International Journal of Computer Games Technology

Joint International Conference on Cyber Games and Interactive Entertainment 2006

Guest Editors: Kevin Kok Wai Wong, Chun Che Fung, and Arnold Depickere

Joint International Conference on Cyber Games and Interactive Entertainment 2006

International Journal of Computer Games Technology

Joint International Conference on Cyber Games and Interactive Entertainment 2006

Guest Editors: Kevin Kok Wai Wong, Chun Che Fung, and Arnold Depickere

Contents

Cyber Games and Interactive Entertainment, Kok Wai Wong, Chun Che Fung, and Arnold Depickere
Volume 2008, Article ID 739041, 2 pages

A Gameplay Definition through Videogame Classification, Damien Djaouti, Julian Alvarez, Jean-Pierre Jessel, Gilles Methel, and Pierre Molinier
Volume 2008, Article ID 470350, 7 pages

Game Play Schemas: From Player Analysis to Adaptive Game Mechanics, Craig A. Lindley and Charlotte C. Sennersten
Volume 2008, Article ID 216784, 7 pages

Story and Recall in First-Person Shooters, Dan Pinchbeck
Volume 2008, Article ID 783231, 7 pages

A Conceptual Framework for the Analysis of First-Person Shooter Audio and its Potential Use for Game Engines, Mark Grimshaw and Gareth Schott
Volume 2008, Article ID 720280, 7 pages

Ambient Games, Revealing a Route to a World Where Work is Play?, Mark Eyles and Roger Eglin
Volume 2008, Article ID 176056, 7 pages

Efficient Terrain Triangulation and Modification Algorithms for Game Applications, Sundar Raman and Zheng Jianmin
Volume 2008, Article ID 316790, 5 pages

Real-Time Optimally Adapting Meshes: Terrain Visualization in Games, Matthew White
Volume 2008, Article ID 753584, 7 pages

Auto Coloring with Enhanced Character Registration, Jie Qiu, Hock Soon Seah, Feng Tian, Quan Chen, Zhongke Wu, and Konstantin Melikhov
Volume 2008, Article ID 135398, 7 pages

Strategic Team AI Path Plans: Probabilistic Pathfinding, Tng C. H. John, Edmond C. Prakash, and Narendra S. Chaudhari
Volume 2008, Article ID 834616, 6 pages

Hierarchical Pathfinding and AI-Based Learning Approach in Strategy Game Design, Le Minh Duc, Amandeep Singh Sidhu, and Narendra S. Chaudhari
Volume 2008, Article ID 873913, 11 pages

A Hybrid Fuzzy ANN System for Agent Adaptation in a First Person Shooter, Abdennour El Rhalibi and Madjid Merabti
Volume 2008, Article ID 432365, 18 pages

Generation of Variations on Theme Music Based on Impressions of Story Scenes Considering Human's Feeling of Music and Stories, Kenkichi Ishizuka and Takehisa Onisawa
Volume 2008, Article ID 281959, 9 pages

A Constraint-Based Approach to Visual Speech for a Mexican-Spanish Talking Head, Oscar Martinez Lazalde, Steve Maddock, and Michael Meredith
Volume 2008, Article ID 412056, 7 pages

Activity Classification for Interactive Game Interfaces, John Darby, Baihua Li, and Nick Costen
Volume 2008, Article ID 751268, 7 pages

A Real-Time Facial Expression Recognition System for Online Games, Ce Zhan, Wanqing Li, Philip Ogunbona, and Farzad Safaei
Volume 2008, Article ID 542918, 7 pages

Perception-Based Filtering for MMOGs, Souad El Merhebi, Jean-Christophe Hoelt, Patrice Torguet, and Jean-Pierre Jessel
Volume 2008, Article ID 243107, 9 pages

A Study of Interaction Patterns and Awareness Design Elements in a Massively Multiplayer Online Game, Tiffany Y. Tang, Cheung Yiu Man, Chu Pok Hang, Lam Shiu Cheuk, Chan Wai Kwong, Yiu Chung Chi, Ho Ka Fai, and Sit Kam
Volume 2008, Article ID 619108, 8 pages

Using a Camera Phone as a Mixed-Reality Laser Cannon, Fadi Chehimi, Paul Coulton, and Reuben Edwards
Volume 2008, Article ID 321708, 6 pages

Using a Mobile Phone as a "Wii-like" Controller for Playing Games on a Large Public Display, Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards
Volume 2008, Article ID 539078, 6 pages

Game Portability Using a Service-Oriented Approach, Ahmed BinSubaih and Steve Maddock
Volume 2008, Article ID 378485, 7 pages

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 739041, 2 pages
doi:10.1155/2008/739041

Editorial
Cyber Games and Interactive Entertainment

Kok Wai Wong, Chun Che Fung, and Arnold Depickere

School of Information Technology, Murdoch University, South Street, Murdoch, 6150 Western Australia, Australia

Correspondence should be addressed to Kok Wai Wong, k.wong@murdoch.edu.au

Received 11 May 2008; Accepted 11 May 2008

Computer games and interactive entertainment have gained much attention recently in the domain of digital media. They are now being applied or used in many areas such as entertainment, education, training, and art. Today, the computer games and interactive entertainment market is highly competitive. In this special issue, all the submissions are invited papers which are extended from original conference papers that were published in the Proceedings of CyberGames 2006 and 2007: International Conference on Games Research and Development, and the Proceedings of the Third Australasian Conference on Interactive Entertainment (IE 2006). This special issue aims to present the latest works on new techniques and applications in the area of cyber games and interactive entertainment. A total of 29 papers have been submitted to this special issue, of which 20 high-quality papers have been accepted after the peer review process. This special issue starts with the first paper entitled "A gameplay definition through videogame classification" by D. Djaouti et al. In their paper, the authors focused mainly on defining game play through some kind of videogame classification. The work presented in this paper is a part of a bigger and global experiment attempting to understand game play better with a study of the nature of videogames. Since game play is an important component in any interactive entertainment design, this work provides some interesting contribution to the field. Still along the area of game play, the second paper is by C. A. Lindley and C. C. Sennersten and is entitled "Game play schemas: from player analysis to adaptive game mechanics." It looks at the use of schema theory and model to understand the cognitive processes underlying game play. This paper examines both the predesigned schema as well as using adaptive game mechanics. In game design, story and narration have become an important area to enhance the game play. The paper by Dan Pinchbeck, "Story and recall in first person shooters," looks into the area of storytelling specifically for the games of first-person shooters (FPSs). With the advancement of technologies, FPS games are able to deliver the high expectation of incorporating a story into the game play. The whole idea of storytelling is to gain the interest of the game players and to perform some indirect control on the players by leading them through the game-play experience. Audio and music have been quite powerful in delivering part of the objective of gaining interest and performing indirect control. The work presented by M. Grimshaw and G. Schott, "A conceptual framework for the analysis of first-person shooter audio and its potential use for game engines," is also useful for achieving such objectives. This paper proposed a new conceptual framework for the design of audio used for developing FPS games. The authors suggested that the framework could allow better immersive experience when playing FPS games. The next paper by M. Eyles and R. Eglin, "Ambient games: revealing a route to a world where work is play," also suggested that audio and music are important in developing good game play. They have introduced a term called "ambient games," which is basically evolved from the concepts of ambient music. They have showed in their paper how to set this concept of ambient games in the technological context. One of the important areas of cyber games and interactive entertainment is graphics. With the advancement of technologies, graphics presented today are much more complex compared to about five years ago. However, researchers are still aiming to find more effective and efficient algorithms for generating better graphics. The paper by S. Raman and Z. Jianmin, "Efficient terrain triangulation and modification algorithms for game applications," presented an efficient terrain generation algorithm. The proposed algorithm is based on constraint conforming Delaunay triangulation. The paper by M. White, "Real-time optimally adapting meshes: terrain visualization in games,"

presented discussions on some of the factors that will affect the terrain visualization in games. It is always challenging to present high-quality scenes through the graphics hardware especially in real-time interactive graphics applications. This paper provided some implementation suggestions that could enhance games and interactive entertainment. Animation is becoming an essential requirement for most interactive entertainment applications, the paper by J. Qiu et al., "Auto coloring with enhanced character registration," presented an autocoloring algorithm using an enhanced-character registration technique. The approach presented can be used for practical animation sequence in achieving high-coloring accuracy.

Kok Wai Wong
Chun Che Fung
Arnold Depickere

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 470350, 7 pages
doi:10.1155/2008/470350

Research Article

A Gameplay Definition through Videogame Classification

Damien Djaouti,[1,2] Julian Alvarez,[1,2] Jean-Pierre Jessel,[1] Gilles Methel,[2] and Pierre Molinier[2]

[1] *IRIT, University of Toulouse III, 118 route de Narbonne, 31062 Toulouse, France*
[2] *LARA, University of Toulouse II, 5 allées Antonio Machado, 31058 Toulouse, France*

Correspondence should be addressed to Damien Djaouti, djaouti@irit.fr

Received 30 September 2007; Accepted 15 February 2008

Recommended by Kok Wai Wong

This paper is part of an experimental approach aimed to raise a videogames classification. Being inspired by the methodology that Propp used for the classification of Russian fairy tales, we have identified recurrent diagrams within rules of videogames, that we called "Gameplay Bricks". The combinations of these different bricks should allow us to represent a classification of all videogames in accordance with their rules. In this article, we will study the nature of these bricks, especially the link they seem to have with two types of game rules: the rules that allow the player to "manipulate" the elements of the game, and the rules defining the "goal" of the game. This study will lead to an hypothesis about the nature of gameplay.

1. INTRODUCTION

This paper is part of a global experimental approach aimed at studying the nature of videogames, in order to try to define what "gameplay" is. The first step of our methodology is to elaborate a classification suited to videogames.

In a simple way, we could consider videogames as an interactive application, entering into interaction with a player.

According to Crawford [1], interactions between a player and a videogame can be perceived as a dialogue: "*A cyclic process in which two active agents alternately (and metaphorically) listen, think, and speak.*"

Through this paper, we aim to focus on the "computer" side of the cycle, in order to analyze the constitutive elements of videogames as pieces of software.

The first target of this approach is to identify formal data, ignoring for now the knowledge and psychological aspects of the player.

The next idea is to study this data in order to deduce a classification of videogames. It should also contribute to the definition of a common language suited to videogame analysis.

We have been inspired by the work of Propp [2] in his study of the Russian fairy tales during the beginning of the twentieth century.

Facing similar problems, such as the impossibility for the researchers of his time to conduct an objective study of the inherent Russian fairy tales mechanisms, Propp used a formal deconstruction.

Starting from a hundred of fairy tales, that he has been analyzed in this way, he has been able to identify recurrent narrative structures which lead him to build a classification of Russian fairy tales.

We have also been influenced by the joint work of Salen and Zimmerman [3], who led us to focus our study on videogames rules : "*Looking at games as rules means looking at games as formal systems, both in the sense that the rules are inner structures that constitute the games and also in the sense that the rules schemas are analytic tools that mathematically dissects games.*"

By isolating the "computer" part of the videogame interaction cycle, we obtain a simple structural diagram (Figure 2) composed of three parts: the "*Input,*" peripheral devices allowing the user to enter choices. These choices are then evaluated by the rules of the "*Compute*" part, in order to produce a "result." This result is finally communicated to the player through the "*Output*" device.

In order to stick to our paradigm, we will focus on the "rules," featured in the "Compute" part, made of software.

According to this approach, we have studied the rules of 588 various videogames. All this data has been indexed in a database called V.E.Ga.S. (*video* & *electronic games studies*).

Our previous researches [4, 5] have shown strong recurrences, on the whole, of videogames rules. These recurrences are exposed in the first part of this article.

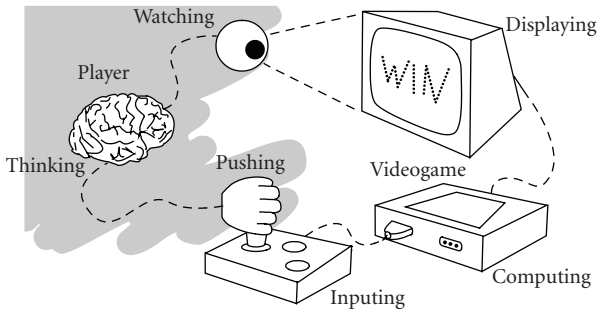


Figure 1: Player and videogame interaction cycle.

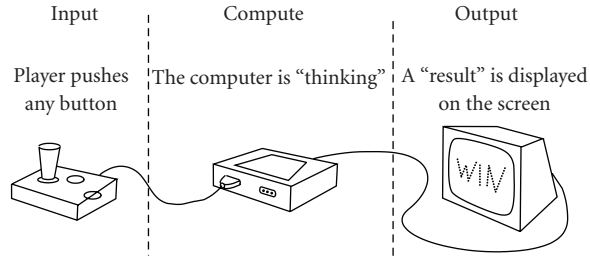


Figure 2: Structural parts of a videogame.

In the second part, we will analyze these recurrences and try to identify the eventual structures that could be related to the gameplay of these videogames.

2. A VIDEOGAME CLASSIFICATION

2.1. Game bricks

In accordance to Propp's methodology, we have developed a tool suited to the indexation and analysis of a large videogames corpus. This quantitative approach should raise eventual recurrent aspects likely to become criteria for a classification.

We based our corpus on as large a period of time as possible, in order to limit the impact of technical evolution on the results we may observe. However, we had to define several limitations to the videogames likely to join our corpus:

(i) single player games only;
(ii) computer games only;
(iii) games based on both audio and graphical output.

The 588 games in our corpus were chosen after an online alphabetical list of videogames titles; however, the great majority of them are "arcade games" or "casual games."

Thanks to our tool, we have proposed a first step for the development of a classification criterion: we have emphasized the "*Game Bricks*" (Figure 3), the "fundamental elements" whose different combinations seem to match the different rules and goals of videogames.

After analysis [7], we noticed that *every "Game brick" corresponds to a "recurrent template" in the rules* of videogames.

For example, two games such as "Pac-man" and "space invaders" features the following rules:

(i) *"If Pac-man collides with Ghost, then destroy Pacman."*
(ii) *"If Spaceship collides with Enemy's shot, then destroy Spaceship."*

We notice a very *strong similarity* between these rules and we can consider, therefore, that they are built on the following template: *"If player element collides with a hostile element, then there is a negative feedback towards the player element."*

This template is then the definition of a "Game brick," namely, the AVOID brick. So far, we have identified ten "Game bricks," all built upon this same principle.

For example, the Game bricks featured in *"Pac-man"* are *"MOVE,"* meaning the player can move an avatar; *"AVOID"* for the ghosts you have to avoid; *"DESTROY"* for the dots you have to eat; and *"MATCH"* because you have to match each dot's spatial position to destroy it.

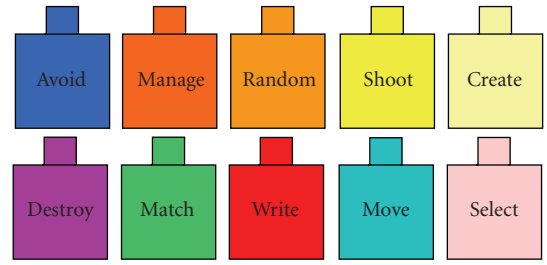


Figure 3: Game bricks discovered as of now. (A side note about the different bricks we have identified: since the paper presenting the first version of "V.E.Ga.S.," some bricks have been modified. You will notice that the bricks TIME and SCORE were removed. The COLLECT brick was merged with DESTROY. The POSITION brick was extended in the form of MATCH. Last but not least, the ANSWER brick was split in two bricks: SELECT and WRITE. More detail on the bricks modifications is presented in [7].)

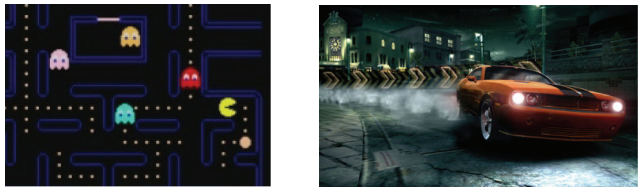
But you can also find these bricks in a racing game like *"Need for Speed"*: MOVE a car, AVOID opponents, and MATCH on checkpoints you have to DESTROY. When reached a checkpoint becomes "out of the game" and is not reachable anymore, so it can be considered "destroyed," just like any dot eaten by Pac-man.

Nevertheless, even within their rules, these two games are different: the movement and thus the "MOVE" brick features two dimensions in "Pac-man," but three in "Need for Speed Carbon"; the number of checkpoints to reach in Need for Speed is much smaller than the number of dots that Pac-man has to swallow; the movement of the elements to avoid is different in each game.

Differences between these two example games are the issue of different implementations of "rule templates" from the bricks they are sharing, but are also due to the use of rules which are not covered by the bricks: in order to obtain an efficient classification we could not make a brick for every existing rule template.

We then had to limit the number of Game bricks, trying to identify the most recurrent rules templates, after a close study of the games in our corpus.

However, the Game bricks are aimed to allow the representation of the diversity of challenges one can find among videogames.



(a) (b)

Figure 4: Pac-man (1980) and need for speed carbon (2006).

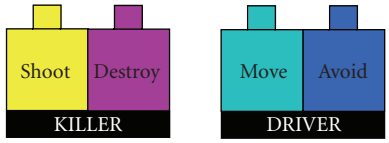


Figure 5: Two identified metabricks.

Besides the recurrent factor, we also took in account the nature of the rule: we have concentrated our efforts on representing the rules related to the actions of the player with the "Game Bricks," meaning we focused on rules related to the game goal and to the means of reaching it.

Being inspired by the works of Koster [8] and Bura [9] who both try to elaborate a grammar of videogames in the shape of diagrams, we have formalized diagrams as definitions for our bricks (*these diagrams are presented in Section 4.1*).

The structure of these templates is based on the "structure of a rule": one or several "*triggering conditions*" (If) associated with one or several *effects* (Then).

The "If...Then" structure of a rule obviously reminds one of the algorithmic scheme used in computer science, as studied in a previous article [7].

2.2. Metabricks

Nevertheless, the number of "total combinations" obtainable with these different bricks is still rather large, but we have noticed that *some couples of bricks were found very often* in a great number of games.

We named those couples of bricks "*Metabricks*" and after the study of games that have one or two of these metabricks, we have given them names that are rather meaningful: MOVE and AVOID became the "*DRIVER*" metabrick, and the association of SHOOT and DESTROY became "*KILLER*."

These "metabricks" seem to us empirically related to the challenges proposed by these games.

Families that have identical metabricks, but also some different bricks seem to present a variation on the same challenge. For example, the families of "Pac-man" and "Frogger" have a difference concerning the DESTROY brick: Pac-man has to swallow dots and thus to destroy them, while the frog has only a busy road to cross.

To summarize, we have identified "Game Bricks" that represent "recurrent rule templates" within videogames. Based on these bricks, we have elaborated a classification that gathers videogames into "families" having identical combinations of "Game bricks."

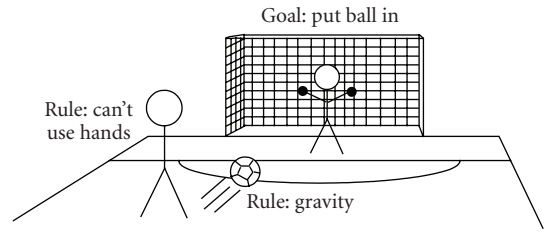


Figure 6: Elements, rules, and goal for soccer.

These families can then be classified through the use of some pairs of bricks, named "MetaBricks."

3. TOPOLOGY OF A GAME

In order to fully analyze the results of our quantitative study, we also have studied the morphology of a videogame in a qualitative way.

We started from the definition of a game according to Salen and Zimmerman [3]: "*An activity with some rules engaged in for an outcome.*"

The authors of "The Rules of Play" consider a game as an activity defined by two elements: the *rules* and the *result*, the latter one coming from a previous goal.

3.1. "Some rules"

If we consider that a videogame takes place in a virtual universe, we can also consider that this universe is composed of several "elements," in the broadest sense.

For example, in soccer, a game that is playable both as a videogame and as a sport, the universe would be composed of elements featured in a match: *players, pitch, goals,* and *ball.*

All these elements are driven by the "rules" of the game, in a similar way that elements from our own universe are driven by physical or behavioral laws.

From a soccer point of view, these rules are the physical rules handling the movement of several elements, like the gravity applied on the ball and the players, but also the game rules specifying that only the goalkeeper is allowed to touch the ball with his hands.

These rules seem to determine a "field of possible actions" that may happen when a soccer match is played. This is what Salen and Zimmerman call the "space of possibility" [3].

3.2. "An outcome"

According to the definition presented previously, a game proposes an outcome. Talking about an outcome implies a judgement of the player performance. But in order to judge, one needs a reference. In a game, the reference is tied to the goal the players have to reach.

For soccer, the goal of the game, identical for each team, is to bring the ball into the goals of the opposing team. The "goals" and "goalkeeper" words are by the way very explicit.

As shown in a previous article [7], we could also consider the goal of the game as a rule, indeed a special one: this rule has to state the end of the game, in others words its outcome, when some conditions are fulfilled.

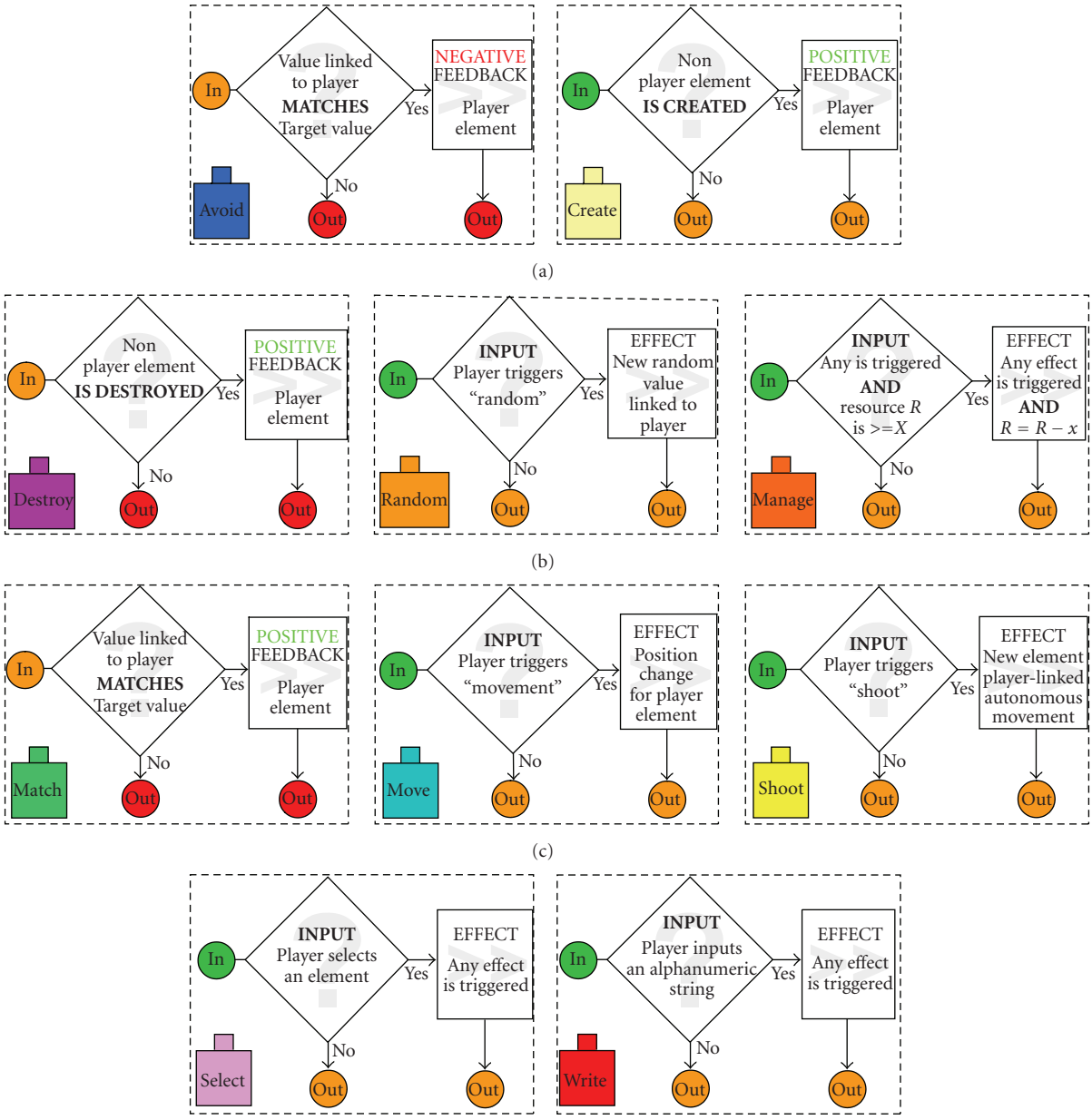


Figure 7

Back to the soccer example, the game is "reset" when the ball enters into one of the goals, and the score of the team featuring the player who shot the ball is increased by one point.

Even though a match ends after 90 minutes, the outcome does not depend only on time: the team with the highest score after 90 minutes of play wins the game.

Hence the judgement allowing the outcome of the game is here tied to the goal of the game, which is to throw the ball into the opposing goal.

3.3. *Different kinds of rules*

If the target of the game is also a part of the game rules, does it means that different "kinds" of rules exists?

The work of Gonzalo Frasca seems to indicate so, in particular his typology of the different kinds of game rules [10].

(i) "*Manipulation rules,*" defining what the player can do in the game.

(ii) "*Goal Rules,*" defining the goal of the game.

(iii) "*Metarules,*" defining how a game can be tuned or modified.

For now, we will put aside "Metarules," which mean that on the whole of videogame rules, we will find some rules related to the definition of a goal, and other rules defining means to reach it.

As different kinds of rules exist, and as "Game bricks" are based upon "rule templates," we can ask the following question:

On what kind of rules are the bricks based on?

4. BRICKS AND GAMEPLAY

4.1. Game + Play = Gameplay?

In order to find which kind of rules the bricks are based on, let us analyze the definition diagrams of each brick.

We notice that the bricks CREATE, DESTROY, RANDOM, MANAGE, MOVE, SHOOT, SELECT, and WRITE all feature a *reference to the videogame's Input within its triggers.*

Please note that these bricks assume that the received inputs are "valid." Hence these "player's inputs" are previously checked by additional mechanisms that are out of the scope of this article.

On the other hand, the AVOID, BLOCK, DESTROY, and MATCH bricks all feature *a feedback within its effects [6]*. (An important note about the use of the word "feedback" in this article: we are aware that within computer science, the terms "negative feedback" and "positive feedback" refer to systems with the ability to automatically correct their actual state. However in the field of game design, "positive feedback" and "negative feedback" refer to the different kinds of "rewards" a game can address to the player. We chose to use the latter definition of these terms in this paper.) This feedback is displayed by the videogame's Output.

We could then divide bricks into two categories, according to whether they feature one or another of these characteristics.

The first category of bricks seems to be based on a principle that one could formulate in the following way: *"to listen to Input and to consequently carry out modifications on game elements."*

The second category would rather correspond to: *"to observe the game elements and to return an evaluation of the quality of modifications made by the first rule category."*

We retrieve here principles close to two types of rules evoked by Frasca: the first category approaches the definition of *"Manipulation rules,"* while the second one seems to be related to *"Goal Rules."*

But, from our point of view, the difference between these two categories of bricks is also tied to the difference between the two words *"Play"* and *"Game."*

Indeed, as the bricks of the first category are related to Input, they can be connected to the word *"Play"*; whereas the bricks of the second category, which are related to the goal and so to the Output, would approach a concept related to the word *"Game."*

Following these observations, we can try to sort the bricks.

The difference between the two bricks categories appears all the more clear by the fact that *they are not in direct relation between each other.*

Indeed, the two categories of *bricks "interact" through the "game elements"*: the *"Play" bricks modify them, and the "Game" bricks observe* the modifications made by the first ones.

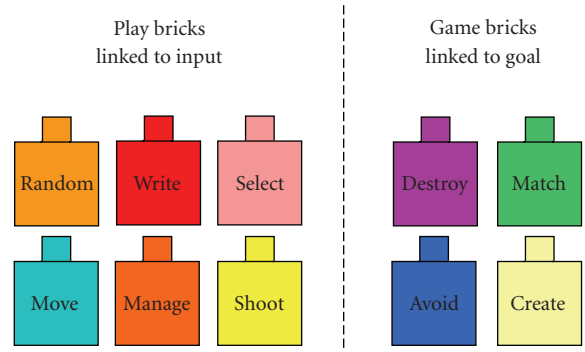


Figure 8: "Play" or "Game" related bricks.

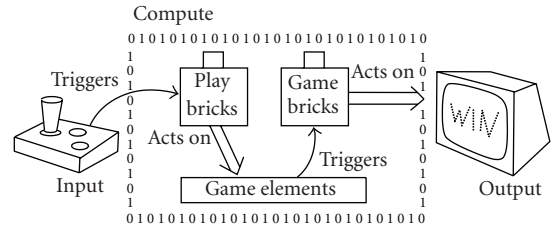


Figure 9: Bricks interaction with input and output.

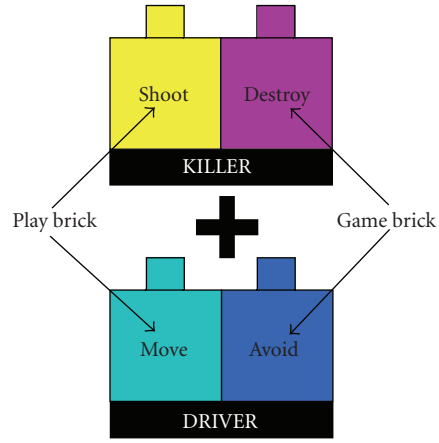


Figure 10: Play brick + Game Brick = Metabrick.

We could finally extend the "videogame structural diagram" (Figure 2) by detailing the "Compute" part, where the rules are located.

Unfortunately, the expression "Game brick" does not seem adequate anymore to refer to our full set of bricks, but only to the subset of bricks from the second category. We must then choose another term, which seems obvious here: we will now refer to the set of 10 identified bricks as *"GamePlay bricks."*

More than a simple name change, this word leads to an important question still looking for a precise answer.

"What is Gameplay?"

Gameplay is empirically seen as a central element within a videogame, and seems closely related to the game quality in the mind of many players.

If the question of its nature appears of capital importance, it is unfortunately a concept which remains to be precisely defined.

Looking for a definition of gameplay, let us synthesize the points studied until now.

We identified a set of recurrences within the rules of videogames, that we named "Gameplay bricks." After analysis, we observe two types of bricks, related to two "kinds of videogame rules."

(i) Rules *listening to Input* and acting on the game elements consequently, named *"Play bricks."*
(ii) Rules observing the state of the game elements and returning to the player an *evaluation of his performance,* named *"Game bricks."*

May the association of "Play bricks" with "Game bricks" be the spirit of gameplay?

A draft answer to this question may come from the two Metabricks presented in Section 2.2, namely, DRIVER and KILLER.

If we analyze them, we notice that *they are composed of a "Play brick" associated to a "Game brick."*

We would say that if the "Game Brick" refers to a *goal to reach*, the "Play Brick" seems to *represent a means* (or a constraint) in order to reach this goal.

For example, DRIVER asks the player to *avoid colliding* with some elements, and allows the player to *move its avatar* in order to do so. In the same way, KILLER asks to *destroy elements*, through the use of projectiles that the player can *shoot or throw*.

As these "Metabricks" represent pairs of "GamePlay bricks," that is, rules templates, which are identified in a large group of games, *our hypothesis about the nature of gameplay seems very promising.*

5. CONCLUSION

Being inspired by the methodology that Propp used for his fairy tales classification, we have started a *quantitative analysis of videogames*.

Propp's methodology leads us to *build a classification based on "recurrent templates of games rules,"* as we identified a set of recurrent rules templates formalized into ten "GamePlay bricks."

According to the work of Frasca, these bricks can be of two kinds.

(i) *"Game"*: if the rule template is directly *related to the goal* of the game, mainly as a feedback within the rule effects.
In this case, the rule is characterized by a trigger based on the state of the game elements, and an effect linked to the *videogame's Output*.
(ii) *"Play"*: if the rule template is *independent from the goal.* The rule is then characterized by a trigger based on the *videogame's Input*, and an effect targeting only the game elements.

We would then state as hypothesis that "Gameplay" is, at least within the videogame rules, *composed of both "Game bricks" and "Play bricks."*

We have then been able to identify pairs of "Gameplay bricks" that have been found recurrently in our games corpus.

We have named these recurrent pairs "Metabricks," as they are composed of "Play brick(s)" associated to "Game brick(s)."

The discovery of "Metabricks," which are the result of pure statistical analysis over a 588 videogames corpus, seems to lean towards a validation of our hypothesis about the nature of gameplay.

However, our corpus of videogames needs to be extended to more games and to more "kinds" of games to fulfill this validation.

Moreover, the expansion of our videogames corpus should lead to the discovery of additional Metabricks: with 4 "Game bricks" and 6 "Play Bricks," numerous new potential metabricks await.

More precisely, the next steps of our study will be based on two complementary approaches.

(i) A "bottom-up" *(qualitative)* approach, which will lead us to pursue the development of an experimental videogame, named *"GamB.A.S" (a first prototype was exposed in a previous article [7])*. The aim of this game is to allow one to observe the interaction between the different kinds of videogame rules, through the ability of enabling/disabling any videogame rule at runtime.
For now, this game only implements rules from the "Gameplay bricks" templates, limiting its videogame generation abilities to quite simplified versions of actual videogames.
(ii) A "top-down" *(quantitative)* approach, which will lead us to pursue the classification of videogames.

We are modifying our classification tool in order to propose a collaborative version of our videogame classification, freely accessible on the Internet.

This improved version adds the possibility to collect and compare a large number of evaluations for each game, in order to minimize the subjectivity introduced during the analysis of videogames.

You might then freely propose, evaluate, or even consult information about any videogame on the following website: http:///www.gameclassification.com/.

ACKNOWLEDGMENTS

The authors wish to thank Jean-Yves Plantec and Martial Bret from the "Iode" company for their point of view on the idea of "bricks," as well as Stéphane Bura, Art Director at "10Tacle Studio," who directed us towards a great number of references. They also wish to offer many thanks to Annika Hammarberg for the translation of this paper from French to English, and Dominic Arsenault for his expert corrections and suggestions. A very special thanks finally goes to Rashid Ghassempouri for his general help and thoughts in our earlier works about the game classification.

REFERENCES

[1] C. Crawford, *Chris Crawford on Game Design*, New Riders, Indianapolis, Ind, USA, 2003.

[2] V. Propp, *Morphologie du conte (1928)*, Seuil, Paris, France, 1970.
[3] K. Salen and E. Zimmerman, *The Rules of Play*, MIT Press, Cambridge, Mass, USA, 2003.
[4] J. Alvarez, D. Djaouti, R. Ghassempouri, J.-P. Jessel, and G. Methel, "V.E.Ga.S.: a tool to study morphology of the video games," in *Proceedings of the International Digital Games Conference (GAMES '06)*, pp. 145–155, Portalegre, Portugal, September 2006.
[5] J. Alvarez, D. Djaouti, R. Ghassempouri, J.-P. Jessel, and G. Methel, "Morphological study of the video games," in *Proceedings of the International Conference on Games Research and Development (CGIE '06)*, pp. 36–43, Perth, Australia, December 2006.
[6] E. Adams and A. Rollings, *Game Architecture and Design*, chapter 3, New Riders, Indianapolis, Ind, USA, 2004.
[7] D. Djaouti, J. Alvarez, J.-P. Jessel, G. Methel, and P. Molinier, "Towards a classification of video games," in *Proceedings of Artificial and Ambient Intelligence Conference (AISB '07)*, Newcastle, UK, April 2007.
[8] R. Koster, "A grammar of gameplay," http://www.theoryoffun.com/grammar/gdc2005.htm.
[9] S. Bura, "A Game Grammar," http://users.skynet.be/bura/diagrams/.
[10] G. Frasca, "Simulation versus narrative: introduction to ludology," in *The Videogame Theory Reader*, pp. 221–236, Routledge, London, UK, 2003.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 216784, 7 pages
doi:10.1155/2008/216784

Review Article

Game Play Schemas: From Player Analysis to Adaptive Game Mechanics

Craig A. Lindley and Charlotte C. Sennersten

Department of Technoculture, Humanities and Planning, Blekinge Technical Institute, Campus Karlshamn, Biblioteksgatan 4, SE-374 35 Karlshamn, Sweden

Correspondence should be addressed to Craig A. Lindley, craig.lindley@bth.se

Received 31 July 2007; Accepted 19 October 2007

Recommended by Kok Wai Wong

Schema theory provides a foundation for the analysis of game play patterns created by players during their interaction with a game. Schema models derived from the analysis of play provide a rich explanatory framework for the cognitive processes underlying game play, as well as detailed hypotheses for the hierarchical structure of pleasures and rewards motivating players. Game engagement is accounted for as a process of schema selection or development, while immersion is explained in terms of levels of attentional demand in schema execution. However, schemas may not only be used to describe play, but might be used actively as cognitive models within a game engine. Predesigned schema models are knowledge representations constituting anticipated or desired learned cognitive outcomes of play. Automated analysis of player schemas and comparison with predesigned target schemas can provide a foundation for a game engine adapting or tuning game mechanics to achieve specific effects of engagement, immersion, and cognitive skill acquisition by players. Hence, schema models may enhance the play experience as well as provide a foundation for achieving explicitly represented pedagogical or therapeutic functions of games.

1. INTRODUCTION

Computer game genres, such as role-playing games (RPGs) and first-person shooters (FPSs), imply particular sets of design features supporting expectations that prospective players have about the nature of the play experience that games support, based upon past experiences with other games in the same genres. When a player first encounters a computer game within an unfamiliar genre, they will, if sufficiently motivated, interact with the game and eventually learn sufficient patterns of interaction to make progress within the game, perhaps eventually completing it. Game play is therefore fundamentally a process of players learning, adapting and improving play skills. Since computer games are predominantly played by the use of very generic interaction technologies (e.g., a keyboard and mouse), learning and adaptation in play are, for the most part, processes of developing cognitive skills focused upon the mechanics of a game and its media realization, based upon an existing general skill set for computer use. Keyboard and mouse operations are mapped onto in-game actions in a game world synthesized by the game software. Learning how to play can therefore be divided into three phases: (1) learning *interaction mechanics*, that is, the basic motor operations required to operate, for example, a keyboard and mouse in a largely unconscious way; (2) learning *interaction semantics*, that is, the simple associative mappings from keyboard and mouse operations to in-game actions (and meta-game actions, such as setting play options, or loading and saving game states); and (3) learning *game play* competence, that is, how to select and perform in-game actions in the context of a current game state in a way that supports progress within a game. Interaction semantics represent a basic level of competence in playing a particular game; these mappings are often carried across different games within a genre and even across genres (e.g., using "w," "a," "s," and "d" keys to move a player character forwards, left, backwards and right, resp.). Learning interaction semantics represents a form of game challenge (in addition to those noted by Rollings and Adams [1]), but once the basic mappings have been learned, they become a largely unconscious foundation for ongoing game play. The focus of learning then shifts to the development of game play competence,

which involves the development of forms of in-game situation awareness and decision making needed to meet the more complex challenges such as those documented by Rollings and Adams [1].

Game play competence involves the ability to (1) decode the audiovisual sensory and perceptual information delivered by the game media (e.g., the computer screen and speakers) into the apprehension of a local situation within the synthesized game world (or game space); (2) evaluate this understanding of the local in-game situation in terms of the overall objectives of play, current goals and tasks, the state of the player character within the game (e.g., capabilities, health, and other statistics), and anticipation of various rewards of playing the game; (3) make decisions about which in-game tactics and action(s) to perform next, based upon the perceived situation and its evaluation; and (4) perform action(s) based upon competence in interaction mechanics and semantics. The details of the cognitive process underlying this repetitive sequence, which could be described as the sense->model->evaluate->plan->act sequence (essentially the same as the sense->model->plan->act structure used to simulate higher-level action control in robots and agents within artificial intelligence research; see http://www.cgie2006.murdoch.edu.au/game_ai.html for extensive references), are the primary higher-level cognitive learning outcome of learning how to play a particular computer game.

The general usefulness of these different aspects of learning in game play relates to the degree to which the knowledge or skills learned may transfer to other contexts. Competence in interaction mechanics is very general, transferring to all contexts within which the same interaction technologies are used; however, the contribution of a particular game to the development of this competence is likely to be very limited, and certainly no greater than other applications using the same interface technology. In fact, a game may be less effective than other applications that are more demanding in terms of knowledge, for example, of keyboard layout, such as word processors. Competence in interaction semantics transfers only to other systems using the same mappings from mechanical interaction operations to in-game actions. This may include many other games, especially those within the same genre but also across genres, depending upon their adoption of implicit or explicit conventions in game-interaction design. However, interaction semantics may be limited in their transferability to other contexts, since contemporary methods of triggering synthetic actions synthesized by a computer game are unlikely to be the same as methods of realizing actions that are not synthesized by a computer.

Game play competence has similar transferability across computer games to competence in interaction semantics, that is, high transferability within a genre but decreasing across genres. However, the potential for transfer of game play competence to contexts other than computer games may be much greater, since similar cognitive processes implementing a sense->model->evaluate->plan->act sequence could apply within those contexts. For example, a flight simulator based upon accurately modeled flight planning and air traffic control procedures may help players to learn how to manage flight planning and air traffic control operations in a real flying context. The key issue here is whether the particular mechanics and design features of the game lead to the development of cognitive structures that can transfer to other contexts. The effectiveness of computer games as situated learning environments (as characterized by [2]) critically depends upon this issue of transfer.

The nature of the cognitive structures underlying game play is not only relevant to knowledge and skill transfer. Those structures are the key to therapeutic applications of game play (e.g., [3]), and in fact are the key to the ability of computer games in all contexts to engage and immerse players and motivate ongoing play. This follows since it is the game play schema driving the situated decision process that determines the nature and timing of emotional rewards motivating play. Hence, a greater understanding of the cognitive structures underlying game play and how motivations and rewards are related to these can aid in better game design in entertainment, pedagogy, and therapy. More than this, it is the central claim in this paper that explicitly modeling those cognitive structures and processes within a computer game engine has the potential to greatly enhance design effectiveness by providing the foundations for the game system itself to guide the development of cognitive structures and control the emotional rewards underlying play.

This paper explores this issue by first considering the cognitive framework for analyzing game play described by Lindley and Sennersten [4]. Methods for conducting analyses of play with a view to identify underlying game play schemas are then described. Based upon this, the paper goes on to consider potential methods by which a computer game system might itself form hypotheses about the schemas underlying the play of a particular game. Finally, we consider some ways in which hypothesized game-play schemas can be used automatically within a computer game system to modify game mechanics as a basis for guiding play and influencing ongoing schema formation and refinement on the part of a player. This work differs significantly from many past projects to create computational players (e.g., see [5]) in that the latter are typically focused on optimal game-play methods that do not need to use computational techniques based upon human play performance. In the case of the work described here, the particular strength and interest of the method are the characterization and explicit representation of the specific algorithmic strategies and cognitive processes of human players, both for analytical purposes and as a foundation for adaptive game mechanics.

2. A COGNITIVE THEORY OF GAME PLAY: TASKS, ATTENTION, SCHEMAS, AND THE PLEASURES OF PLAY

Lindley and Sennersten [4] present a theory of the underlying cognitive systems involved in game play based upon schema theory and attention theory. Schemas are cognitive structures that link declarative (or factual) and procedural (or performative) knowledge together in patterns that facilitate comprehension and the manifestation of appropriate actions within a context. While the taxonomical structures

of semantic or declarative memory are comprised of object classes together with associated features and arranged in subclass/superclass hierarchies, the elements of schemas are associated by observed contiguity, sequencing, and grouping in space and/or time [6]. Schemas can refer to declarative knowledge and taxonomical types with their features and relationships, and integrate these with decision processes. Schemas include *scripts* for the understanding and enacting of behavioral patterns and routines, a classic example being Schank and Abelson's [7] example of the *restaurant script* that includes a structure of elements for entering a restaurant, sitting down, ordering food, eating, conversing, paying the bill, leaving, and so on. Scripts, as structures used for both comprehension and behavior generation, represent a structure of cognitive functions that may include cognitive resources, perceptual interpretations and preconditions, decision processes, attention management, and responsive motor actions. *Story schemas* are patterns representing a structure of understandable elements that must occur to make stories comprehensible. The presence of story schemas in the cognitive systems of storytellers, listeners, readers, or viewers of stories allows stories to be told and to be comprehended, including the inference of missing information. If a story deviates too far from a known schema, it will not be perceived as a coherent story. Script and story schemas are concerned with structures of both space and time, while *scenes* are schemas representing spatial structures, such as the layout of a house, a picture or an area of a city.

While schemas have been interpreted in many different ways, here a *game play schema* is understood as a cognitive structure for orchestrating the various cognitive resources required to generate motor outputs of game play in response to the ongoing perception of an unfolding game. A game play schema is therefore the structure and algorithm determining the management of attentional and other cognitive, perceptual, and motor resources required to realize the tasks involved in game play. Examples of types of game-play schemas described by Lindley and Sennersten [4] include story scripts for understanding high level narrative structures designed into games, and scripts for the combative engagement of an enemy, exploring a labyrinth, interacting with a trader nonplayer character, and negotiating and carrying out quests.

Attention theory provides an account of the energetic resources available to cognition, together with principles for the distribution of energy (or attention) to the cognitive resources that use (or manifest) it. Attention theory addresses issues of attentional focus, management of attention (including attentional selection), and the allocation of cognitive resources to cognitive tasks. Ongoing research is addressing the question of the detailed operation of attentional mechanisms, including questions such as the degree to which attentional capacity is specific to specific cognitive resources (or modes) or sharable among resources according to demand, and the stage of processing of perceptual information at which perceptual information is selected for attentional priority. Schemas can be regarded as mechanisms or algorithms that, among other functions, determine the allocation of attention to cognitive tasks.

In the context of game play, attention and the operation of game play schemas are driven by hierarchical goals that set tasks for a player. Goals include those intended by designers and those created by players as allowed by a game design. A hierarchical decomposition of game play goals might at a high level include the completion of a game, which decomposes into the subgoals of finishing each of its levels, each of which in turn decomposes into goals of completing a series of game challenges (and other tasks invented by the player).

We hypothesize that this hierarchical goal structure is mirrored in a hierarchical structure of schemas within a player's cognitive system, where a schema is an algorithm for completing a particular goal or subgoal. As argued by Lindley and Sennersten [4], this schema structure is fundamental to many aspects of the pleasures and motivating factors behind play. These include the pleasures of the following.

(i) *Effectance* which is a basic feeling of empowerment created when an action of a player results in a response from the game system [8]. The cause-effect relationships underlying effectance are a fundamental premise of goal-oriented schemas for action.

(ii) *Closures* at different hierarchical levels (as described by Holopainen and Meyers [9]), where a closure is interpreted here as the completion of the algorithm constituted by a play schema. Closures may involve completion of expected outcomes and resolution of dramatic tensions, corresponding to the completion of cycles of suspense and relief identified by Klimmt [8]. A distinction must be made here between the intrinsic pleasures of schema completion and more complex emotional experience and rewards due to fictional identification within the game world (see the point below regarding episodes).

(iii) *Achievement* of in-game tasks which is rewarding due to the *displacement* of a player's identity into their character [9], this being a matter of *imaginative immersion* as described by Ermi and Mäyrä [10]. Achievement-oriented reward is a more specific form of reward than mere closure, since it is associated with the completion of schemas by the achievement of specific goals.

(iv) More complex forms of enjoyment in game tasks regarded as *episodes* [8] following from imaginative displacement into the game world. Enjoyment within episodes may include the excitement of possible action, the pleasures of curiosity and discovery, the pleasures of experiencing negative emotions of suspense followed by the transference of arousal to an ecstatic experience when the challenge creating the anxiety of suspense is overcome, and enhanced self-esteem. Schemas offer greater discrimination of the pleasures involved in episodes by allowing different forms of episodes to be modeled as different schema patterns having a complex substructure with corresponding emotional effects (e.g., different scripts for solving mysteries, combat, exploration, trading, and quest negotiation).

(v) *Escape* to an alternative reality provided by the fictional world represented by a game [8] and facilitated by imaginative displacement. Players have the pleasure of being able to experience new objects, actions, social interactions, and experiences at no risk. These vicarious experiences can help players to cope with felt frustrations and deficiencies in their everyday lives, a process both of catharsis and of perception of increased competence and relevance. Schemas for stories facilitate displacement, while many additional schema forms provide the foundations for comprehension of the events within the fictional world and provide mechanisms for projection of the player's sense of self into the fiction.

(vi) Achievement of a sense of *flow* [11] in game play, this being a state at the boundaries between engagement and immersion, of being totally absorbed in meeting a constantly unfolding challenge. We hypothesize that the flow state is associated with attentional demand, in particular occurring when schema execution demands attentional resources above a level that would result in player boredom and below a level that would result in excessive difficulty and consequent frustration.

Schema theory therefore has the potential to provide both an explanation of the decision and operational processes underlying game play and an explanation of the detailed reward and motivation factors behind play. Validating this potential requires detailed study of play resulting in the development of empirically validated hypotheses about the detailed structure and functionality of game-play schemas, for individual players and across groups of players.

3. METHODOLOGIES FOR IDENTIFYING GAME PLAY SCHEMAS

Identification of game play schemas is a knowledge acquisition and representation process. Our current methodology for doing this includes analysis of the design features of test games, logging of player key strokes and mouse movements, recording of the screen history of play, eyetracking data showing the locus and dynamics of player gaze behavior, and think-aloud protocols to gain some insight into the player's conscious experience of play and its decision processes. Analysis of this data then proceeds by a process of detailed analysis of individual play sessions in order to identify different play modes and abstract hypothetical underlying game play schemas. This in itself is a complex process that may begin with cognitive task analysis (CTA, see http://mentalmodels.mitre.org/cog_eng/ce_methods_I.htm), but must end with a detailed cognitive explanation of the decision processes involved in terms of basic cognitive functions. Statistical patterns of play interaction (mouse moves, key strokes, and eye movements) that may correlate with the presence and execution of specific game play schemas are then identified. This requires the separation of an analysis dataset from which schema models and initial statistical distributions are derived from a test dataset that can then be used to validate those schema models. This sequence is iterated in order to refine the identified schema models.

The design features of the games used within these studies are crucial. Hence, an initial analysis of the selected games must be made in order to identify their general features. The iterative process of refining and validating hypothetical game play schemas must also involve the creation of purpose-specific test games or levels, this being done by level editing and modding (i.e., modification of off-the-shelf games, potentially including their media content and scripted behavior). It is also possible to implement a hypothetical schema to create a computational player and to test the resulting game play interactions with actual player interaction as another method of validating a schema hypothesis. As noted by Lindley and Sennersten [4], a CTA provides the first approximation description of a game play schema, but a CTA is also heavily determined by the language and cultural constructs of the observer. The phenomenologically meaningful terms of a CTA may have to be further analyzed to account for the ways that those high-level constructs are actually realized by underlying neurophysiological mechanisms, and this mapping could involve different parsings of functional units at the CTA and neurophysiological levels. Hence, a game play schema might be described at different levels of abstraction or from different interpretation perspectives, some being meaningful in terms of the subjective languages of task performance (e.g., the terms of self-reported task performance) or CTA and others in terms of implementational neurophysiology that may have a very different structure and functional decomposition than that of more linguistically conditioned accounts.

The choice of the level of abstraction in game play schema descriptions may depend upon the purpose of the analysis. More importantly, however, it may be that distinctive statistical profiles can be associated with schema characterizations at an optimal level of abstraction; more abstract schema descriptions may be too general to have any statistical discrimination between them, while more detailed descriptions may involve details that cannot be correlated with statistical groups. Hence, an important ongoing task is the statistical validation of suitable levels of schema description. It is yet to be determined how consistent the level of description needs to be, across game genres, games within genres, different kinds of players, different players within those types, and different play sessions for the same player. It is hoped, however, that applying this methodology will result in statistical profiles uniquely associating player types and game design feature sets with distinctive statistical distributions of interaction primitives at the level of interaction semantics that indicate specific hypothetical game play schemas (or sets of schemas) within the cognitive systems of those players. This is a large undertaking (and in fact endless, as game design continues to evolve) that must be approached incrementally by focusing upon specific genres, games, and design feature subsets.

Questions of levels of abstraction and also of higher-level structures also apply to interaction primitives. Basic interactions implement game moves at the semantic level. However, the presence of specific schemas may be indicated by specific sequences or clusters of interaction semantics, rather than, or in combination with, their frequency. Different play modes,

such as setting options versus game play commands directed towards achieving in-game goals (i.e., game moves) can often be distinguished by specific discrete interaction primitives, such as hitting an escape key. However, manual interpretation of play and the formation of schema hypotheses based upon this are crucial for defining criteria for distinguishing between the presence of different schemas that involve the same or similar interaction primitives.

To illustrate this discussion, a hypothetical schema can be described based upon observations of play of the role playing game *Neverwinter Nights*. *Neverwinter Nights* is a third-person point of view game in which the player has a primary in-game character and this character can gain a number of companions in order to form a team, also controlled by the player. Within the game world there are many underground labyrinths consisting of rooms and chambers connected by passages. Rooms and passages often have doors and the labyrinths in general contain threats such as monsters and traps, nonplayer characters that may be friendly or hostile depending upon how the player character interacts with them, powerups and various treasures. The play patterns observed for this example occurred within a period of the game during which the player is intended (by the game designers) to be acting to achieve a number of higher-level story goals predesigned into the game; in particular, the player is on a quest to find four specific creatures that are the key to creat an antidote for a plague and each of which is hidden somewhere within its own labyrinth. Each labyrinth has a similar abstract structure and distribution of game challenges, with differences in its thematic realization. This leads to a style of game play that manifests highly repetitive patterns of interaction and decision making. The schema, expressed in this case in a kind of high-level and informal pseudocode, is a hypothesis about (part of) the underlying algorithm responsible for manifesting these repetitive patterns as the player character and team move through the labyrinths.

The question of the level of abstraction involved is illustrated by considering a significant number of possible subtasks and additional tasks that are not represented in the above description: Check health bar for $1, \ldots, N$ characters, Check for treasure/items to pick up, Check item attributes/quest relevance, Select navigation waypoints for movement, Avoid enemies during retreat, Tweak group member positions, Bring back strays, Check status of quests, Talk with NPCs, Accept/reject quests, Check minimap window, Reconfigure inventory, Reconfigure equipped items, Select level up options, and so forth.

A complete schema description must include all possible subschemas and include a way of representing the operation of simultaneous parallel schemas, their relative priority, and the principles for switching from one schema to another. The detail involved can be high. For example, the detailed description of a subtask such as "check map window" must include an account of exactly what it is that is being looked for in the map window, how the data is to be interpreted, and some kind of representation for the outcome of the minimap check (e.g., a decision about being lost and/or activating a goal-related reorientation subschema).

```
1. Stop at Closed Door
2. Check health of party
        if >1 party member low, then:
                Rest Party
                Resummon summoned creature
        else
          if 1 party member low, then:
                if lots of healing potions, then:
                        administer healing
                else
                        Rest Party
                        Resummon summoned creature
3. Enter combat configuration
4. Open door and enter room
5. If there is an enemy
        Select target
        Monitor health of party until enemy defeated
        if >1 party member has low health, then:
                Run away
                Rest Party
                Resummon summoned creature
                Go back to step 3
        else
          if 1 party member low, then:
                if lots of healing potions, then:
                        administer healing
                else
                        Run away
                        Rest Party
                        Resummon summoned creature
                        Go back to step 3
6. If enemy remains, go to step 5
7. Check for traps ...
... etc. ...
```

Algorithm 1

4. AUTOMATED IDENTIFICATION OF GAME PLAY SCHEMAS AND SCHEMA-BASED ADAPTATION OF GAME MECHANICS

The intended outcome of schema analysis over significant numbers of players and play sessions is a probabilistic profile of the frequencies, clusters, and/or sequences of semantic interaction primitives (game moves) associated with different types of underlying game play schemas for a specific game (i.e., its design features). If such a set of statistical profiles is available, it may be possible to use the profiles for automated identification of the schemas of particular players/play sessions. It is possible to automatically record (or log) interaction semantics for a particular player during a particular play session or across different play sessions. This will result in a count of the absolute frequency of each type of semantic interaction primitive used by that player, which can be turned into a relative frequency by subdivision with the total count of interaction primitives. This might be used (in addition to specific commands that indicate changes in play mode) to match against a database of statistical profiles of different game play schemas in order to derive a probabilistic

hypothesis about the likelihood that specific known schemas are underlying play. Here, we hypothesize that poor overall correlations are likely to indicate the presence of previously uncharacterized schemas, which ideally should be returned to a central schema repository for the game for analysis and new schema description development and distribution.

As described above, game play schemas represent the significant learning outcomes of game play and also encapsulate various rewards of play. An explicit representation of desired and observed game play schemas within a game system constitutes a knowledge base that can potentially be used automatically for a variety of purposes by the game system.

Schema representation and mapping can be used for automated monitoring of a game design as a method of validating the design in terms of player satisfaction. Since a schema includes various points of player reward and presents a time structure for the emotional experience of a game, the schema indicates a hypothesis about the nature of the emotional experience of the player. Different players may prefer different forms of satisfaction. Monitoring schemas and schema execution may indicate which forms of satisfaction particular players are seeking. It may also provide a foundation for determining when a player is not achieving enough satisfaction (based upon criteria that may be derived from a player's history of play, since different players may have different demands in terms of the nature and intensity of rewards). This may be because a game is too easy or too difficult, in which case the game mechanics and parameters determining difficulty for a specific schema can be modified to achieve a better subjective experience. It may also be that a player has not discovered those elements of a schema needed in order for its execution to result in a satisfying experience, in which case the game system model of the player schema related to a target schema might be used to change the game mechanics, for example, to dynamically adapt a level design or to introduce instructional material (perhaps by spawning a suitably informed NPC) to lead the player to actions (such as going to a specific training scenario) that result in gaining the appropriate skills. In effect, this can amount to more efficient and dynamic use of in-game tutorials together with an automation of the normal processes of game tuning carried out manually during game testing prior to release, but having the advantage of being tuned to specific players rather than a group of commercial testers.

Schema descriptions can also be used to explore the effectiveness of a game design in realizing designers' intentions. Simple observations may indicate basic design failures, such as the visual design of interactive elements leading them to be too unobtrusive within the game space for players to notice. However, schema descriptions may show deeper and less obvious problems, such as design features leading too soon to limited modes of play that reward players too much for play patterns that are developed very quickly, discouraging them from exploring a game enough to discover other aspects of its mechanics. As with the other examples presented here, actively using these models within a game system allows the mechanics to be varied for individual players, instead of providing a single solution that is supposed to accommodate everyone.

Monitoring schema formation can also result in automated detection of the degree to which a game design is achieving *emergent game play* (see [12]) where the design rather loosely constrains the nature of the play experience. In this case, poor correlations with known schemas may be a positive indicator of emergent play. Conversely, design features may be selected that are compatible with a broad set of known schemas representing very different play styles, ensuring that a design accommodates a wide variety of play styles, a space within which players have a lot of freedom to create patterns of interaction.

A major use of explicit schema representations may lay in pedagogical or therapeutic functions of game play. In this case, target schemas may not be initially derived from game play but from the target application domains for learning or training. For example, in a military application, observation of tactical decision making in the field could support the development of schema descriptions for tactical decision making. A game for tactical training should then encourage players to preferentially develop the same or, functionally, similar schemas. The effectiveness of the design of a game intended for tactical training can then be assessed by comparing the schemas of players derived from observation of their play patterns with those of operational tacticians. This may be a great advantage compared to assessing performance outcomes, since performance outcomes alone only indicate how a player has mastered a game system, with no indication of how well the mastery of the game will transfer to an application domain. The schema description is an explicit representation of the cognitive capabilities that facilitate operational competence, thereby having much greater transfer potential from the game to the target application environment. Moreover, aspects of the operational schemas that cannot be facilitated by game design provide an explicit representation of the limits of transfer that may then be used to appropriately focus on supplementary training.

Just as in the case of tuning game mechanics for player satisfaction, explicit schema representations and monitoring of player schemas can be used to adapt game mechanics to achieve pedagogical or therapeutic outcomes. For example, a game designed to train players to achieve batter spatial navigation skills might present an initial diagnostic level involving a comparatively complex navigation task based upon a variety of cues, such as verbal descriptions, minimaps, distance cues, and local cues like footprints and vehicle tracks. Based upon which cues players use, ongoing levels can reduce or exclude cues that are already taken into account and emphasise neglected cues to encourage the development of broader attention patterns.

5. CONCLUSION

This paper has described an approach to the analysis of game play based upon schema theory and attention theory. An empirically based method has been described as a basis for identifying and validating hypothetical game play schemas. Automated schema recognition and the potential uses of explicit schema representations within game systems have been explored. This approach provides for explicit modeling of the

cognitive systems and processes underlying game play, both for analytical studies of play and as a potential implementation mechanism for adaptive games. Work on the analysis of games using this approach is ongoing. It is hoped that the results of this work will provide the foundations for future implementation of schema-based adaptive game systems.

ACKNOWLEDGMENTS

This paper has been written in the context of the FUGA (FUn of GAming) EU research project, and a collaboration with the *Swedish Defence Research Agency* (FOI). We thank our colleagues for many stimulating inputs to this project.

REFERENCES

[1] A. Rollings and E. Adams, *Andrew Rollings and Ernest Adams on Game Design*, New Riders, Indianapolis, Ind, USA, 2003.

[2] J. P. Gee, *What Video Games Have to Teach Us About Learning and Literacy*, Palgrave Macmillan, New York, NY, USA, 2003.

[3] G. Robillard, S. Bouchard, T. Fournier, and P. Renaud, "Anxiety and presence during VR immersion: a comparative study of the reactions of phobic and non-phobic participants in therapeutic virtual environments derived from computer games," *Cyberpsychology & Behavior*, vol. 6, no. 5, pp. 467–476, 2003.

[4] C. A. Lindley and C. C. Sennersten, "A cognitive framework for the analysis of game play," in *Proceedings of the 28th Annual Conference of the Cognitive Science Society: Workshop on the Cognitive Science of Games and Game Play (CogSci '06)*, Vancouver, Canada, July 2006.

[5] H. J. van den Herik, Y. Björnsson, and N. S. Netanyahu, "Computers and games," in *Proceedings of the 4th International Conference (CG '04)*, Ramat-Gan, Israel, July 2004, Revised Papers. Lecture Notes in Computer Science 3846 Springer 2006.

[6] J. M. Mandler, *Stories, Scripts and Scenes: Aspects of Schema Theory*, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1984.

[7] R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillsdale, NJ, USA, 1977.

[8] C. Klimmt, "Dimensions and determinants of the enjoyment of playing digital games: a thrre-level model," in *Proceedings of Level Up: Digital Games Research Conference*, M. Copier and J. Raessens, Eds., pp. 246–257, Utrecht, The Netherlands, November 2003.

[9] J. Holopainen and S. Meyers, "Neuropsychology and Game Design," Consciousness Reframed III, Newport, Walse, UK, http://www.stephan.com/NeuroBio.html, May, 2006.

[10] L. Ermi and F. Mäyrä, "Fundamental components of the gameplay experience: analysing immersion," in *Proceedings of the Digital Games Research Association Conference, Changing Views: Worlds in Play (DIGRA '05)*, S. de Castell and J. Jenson, Eds., pp. 17–25, Vancouver, BC, Canada, June 2005.

[11] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper Perennial, New York, NY, USA, Reproduction edition, 1991.

[12] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*, MIT Press, Cambridge, Mass, USA, 2004.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 783231, 7 pages
doi:10.1155/2008/783231

Research Article

Story and Recall in First-Person Shooters

Dan Pinchbeck

University of Portsmouth, Eldon Building, Portsmouth, Hampshire PO1 2DJ, UK

Correspondence should be addressed to Dan Pinchbeck, dan.pinchbeck@port.ac.uk

Received 28 September 2007; Accepted 15 February 2008

Recommended by Kok Wai Wong

Story has traditionally been seen as something separate to gameplay—frequently relegated to an afterthought or epiphenomenon. Nevertheless, in the FPS genre there has been something of a renaissance in the notion of the story-driven title. Partially, this is due to advances in technology enabling a greater capacity for distributed storytelling and a better integration of story and gameplay. However, what has been underrecognised is the dynamic, epistemological, and psychological impact of story and story elements upon player behaviour. It is argued here that there is evidence that story may have a direct influence upon cognitive operations. Specifically, evidence is presented that it appears to demonstrate that games with highly visible, detailed stories may assist players in recalling and ordering their experiences. If story does, indeed, have a more direct influence, then it is clearly a more powerful and immediate tool in game design than either simply reward system or golden thread.

1. INTRODUCTION

In our recent paper [1], we discussed a simple study designed to offer evidence of one of the ways in which nonludically significant devices within an FPS game have a direct impact upon how the experience of the game is recalled and reported. Subjects played one of two games with very different levels of visibility and importance of story and then undertook a semistructured interview based around four major aspects: plot, character, avatar, and world. The research focus was whether subjects appeared to utilise story as a framework for their recall.

Narrative and games have a rather turbulent historical relationship, particularly in academic circles, but it is not our intention to add to volume of writing on this subject here. Thus, rather than entering into a debate about the relative narrativity of games, we will concern ourselves with the functional operation of story in FPS systems and, in particular, how they might relate to memory.

Firstly, we will offer an alternate way of conceptualising story information in games that coopts both Barthes' atomisation of narrative [2] and Carr's appropriation of the protonarrative [3]. Secondly, we will briefly introduce the notion of narrative psychology and the claims that there exist innate story-like structures in cognition that would predispose subjects towards the favoring of story-like sequences, the interpretation of sequences as stories, or better recall of sequences with highly apparent story-like structures.

2. STORY IN GAMES AS PROTONARRATIVE NETWORK

Barthes argues for four fundamental units that act as the building blocks of all narrative [2, pages 79–124]. These units are divided into functions, which relay action, and indices which relate to abstract, atmospheric, or psychological notions. For example, the unit "He wrote" would be classed as a function as it conveys an irreducible action, whereas "He was tense" falls into the category of indices. Both categories may be further defined: functions according to their relative importance and impact, and indices according to their level of abstraction or specifity.

Barthes classes functions as either cardinal, which are both consecutive and consequential, or catalyst, which are only consecutive. In other words, cardinal functions are critical to narrative progression, whereas catalysts may be crucial to the telling of the story but their omission will not affect the basic structure of the story itself. True indices refer to "the character of a narrative agent" [2, page 96] such as an emotion, mood, or atmosphere, whilst the other subclasses, informants, locate within the temporal environment of the narrative. Barthes argues that everything within a narrative is essentially constructed from these four classes of objects.

In itself, this taxonomy is important because it enables a structural, rather than semantic, classification to occur when approaching narratives, asking questions such as whether there is a dearth of catalyst functions (suggesting a lean, reportage style of text), or large numbers of informants presen (perhaps prompting a higher level of trust in the reliability of the narrative).

The idea of functional units is extremely powerful because they operate far below the threshold of a fully formed narrative situation that has been so problematically applied to games. Conceptually, it is not difficult to conceive of a situation whereby as Jenkins notes "Narrative can also enter games on the level of localized incident, or what I am calling micronarratives" [4, page 125]. However, his conceptualization of micronarrative is problematic as it just seems to lower the threshold beyond which a sequence can be defined as a narrative, which rather misses the point of the ludologists issues with their structure rather than their scale [5, 6]. More useful is Carr's notion of protonarratives.

Carr also notes that narratives are composed of isolatable units that, whilst not containing explicit causal sequences, have a form of interpretative predisposition hardwired into them [3]. In other words, we can understand a network of units that, when perceived, is likely to lend themselves to one particular interpretation over another. Bartlett famously conducted a study, whereby native American myths with nonwestern narrative structures were converted in memory by Western subjects to yield more conformist and recognisable Western narratives [7, pages 64–94]. Between the preexisting interpretative structures of the subject and a network of protonarrative units that predisposes a particular interpretative outcome, a story is formed. Thus, for the remainder of this paper, when we speak of stories, it is the protonarrative network with its predetermined relationships we are referring to.

This conceptualization of story neatly sidesteps the narrativity debate by not requiring games themselves to be narrative objects, but simply to contain a set of objects at least some of which have predetermined semantic as well as ludic relationships. Protonarrative networks may contain sequences, but it is just as likely that the colocation of their constituent elements will be enough to trigger a particular interpretation by the player. There are very clear links here to schema theory.

3. SCHEMA, STORY, AND GAMES

Schema theory essentially posits a set of inbuilt and learned mental architectures that hold generalised situational knowledge. Schank and Abelson's contextual dependency theory [8] is schematic at root such as Minsky's frames: "a data-structure for representing a stereotyped situation" [9, page 1]. The basic notion of schema is that once enough cues have been received to trigger the schematic response; the following operations are essentially put through this filter.

Thus, not only may we infer a story schema from Bartlett 's study, but we can also propose schema for media experience and gaming. Indeed, Ijsselsteijn (2003) argues that learned schemata are fundamental factors in users' experiences of media:

From the anecdotal evidence accumulating throughout media history, it becomes clear that people's responses to media are not a linear product of the extent of sensory information that the medium provides but are very much shaped by people's previous experience with and expectations towards media. It would seem a little odd to us now if people should panic and run out of a movie theatre at the sight of an approaching train on the screen. This is because our *media schemata*, or knowledge representations of what media are, and are capable of, tell us what to expect from mediated experiences, including the perceptual tricks that cinema or VR can play on us. (2003:37).

When schemata are triggered, they adjust the interpretation of further signals, making other schemata more or less likely to fire in turn: priming occurs. For example, closure is generally agreed to be a fundamental, even unavoidable aspect of narrative, to the extent that Kermode wonders, "Why does it require a more strenuous effort to believe that a narrative lacks coherence than to believe that somehow, if one could only find it, it doesn't?" [10, page 53]. Thus, when the schema for story fires, we could postulate that it increases the expectations of closure and following the type of interpretive activity that may occur. Equally, once a situation crosses the threshold to be identified as a first-person shooter, the shooter schema fires and predisposes a certain type of perceptual activity and action. In a pilot study, Pinchbeck et al. [11] noted that experienced FPS players tended to centralize their gaze and use the mouse to visually explore the environment, whereas inexperienced players tended to keep the mouse static and allow their gaze to move around the screen. A possible explanation of this disparity in visual behaviour is that experienced players know that shots, when they need to be fired, will hit whatever is central on the screen. Thus, it is advantageous to synchronise the acts of visual exploration and aiming.

Schemata offer a model by which a network of units would trigger a particular interpretative frame. In other words, all a game needs to do is to contain a suitable network of protonarrative objects, and a player will tend towards a story-like interpretation of their play. This offers a functional bridge between Juul's "real rules and fictional worlds" [12, page 163]. A story interpretation, therefore, becomes, as Rein and Schon describe "one of a class of *framing* procedures, that is, strategies for organising and deriving solutions for problems" [13], or "an organising principle for human action." [14]. In other words, a protonarrative network which triggers a story schema to fire as the primary interpretative device may help players to understand what occurs during the game session It should also be noted that there is no contradiction or complication arising from both story and shooter schema firing simultaneously. The experience of playing, Bioshock (2007) or S.T.A.L.K.E.R. (2007) games which invests so much, so clearly, in their stories, is clear evidence of this.

This paper is primarily concerned with the simple question of whether a strong story helps players remember more of their gameplay experience and in better detail. This

begs the question of why we might expect this to be the case, and that requires us to consider the relationship between memory and story in a little more detail.

4. STORIES AND MEMORY

In his classic study, Tulving proposed a distinction between two forms of memory:

> "Episodic memory receives and stores information about temporally dated episodes or events, and temporal-spatial relations among these events. A perceptual event—is always stored in terms of its autobiographical reference to the already existing contents of the episodic memory store. Semantic memory is the memory necessary for the use of language. It is a mental thesaurus, organized knowledge a person possesses about words and other verbal symbols, their meaning and referents, about relations among them, and about rules, formulas, and algorithms for the manipulation of these symbols, concepts, and relations." [15, pages 385–387].

Eysenck and Keane [16, page 165] note that there has been some controversies whether this distinction actually obfuscates a unitary process, but it is the natural fusion of episodic memory and schema that are of interest here. In particular, the autobiographical and temporal aspects of episodic memory have very clear conceptual relationships to the notion of story schema and, indeed, narrative psychology has made much of this.

Conceptually, there is only a short step to be made from an autobiographical memory, where a sense of coherent self relative to a sequence of events is held by an organism, to the idea that our inner experience is a form of story in itself—though at this point we will revert to the term narrative in keeping with the literature and to avoid confusion with our ludically orientated definition of story used elsewhere. Robinson and Hawpe make this point, for example, "Experience does not automatically assume narrative form. Rather, it is in reflecting on experience that we construct stories" [13, page 111]. This would seem to suggest that whilst episodic memory in itself needs not be explicitly narrative, any reflection upon it is. Further, Crossley argues that by planning our lives and ourselves, we engage in a filtering process, thus, creating an ongoing narrative from the nonlinear, frequently noncausal complexity of life [17]. Equally, our own actions and responses, including memories, are filtered and represented to form the self.

Indeed, the self may be seen as the result, illusory or not, of predictable responses to environmental situations. In other words, a sense of coherence in response suggests a coherent, locatable self at the centre of a complex and shifting world. Thus, the inward projection of a self as distinct from the environment, tied up in the process of distal attribution, or reality inferral [18], requires a temporal sequence of coherence in order to logically maintain itself. In other words, without a sense that the self that is being postulated existed prior to the current experience, and without the sense that past and current experiences fall along a single trajectory of ongoing experience, the construct cannot be maintained.

The autobiographical function of episodic memory may, then, not simply be to enable the storage, recovery, and implementation of representations of prior events and body states, but also to enable the illusion of a singular, unitary self to exist in the first place. Just as the psychological processes surrounding presence, vection, and so forth, all enable the organism to separate itself from the environmental field, so episodic memory anchors it to its own developmental past. If the formation of causal sequences in relation to a stable perspective is fundamental to the sense of self, then, narrative is given a very core position in our subjective being indeed. Bruner, for example, argues for a "protolinguistic readiness for narrative organisation and discourse" [19, page 80], whilst Nath [20] contends that narrative is a critical component of assembling a subjective stance and thus crucial to any experience. Underlying these claims is, however, a rather weak and inclusive definition of narrative, more an "assimilating structure" [21, page 91] than a represented causal sequence. We are reminded of Juul's point that in the narrative/ludology debate, the narrativists were characterised by a very inclusive definition of narratives, whereas the ludologists demanded a much less inclusive one [12, pages 156–159].

Once again, schema theory offers a middle ground. We do appear to be, at least given the evidence of their ubiquity, story-telling animals as Schank has argued [22]. Stories are common and powerful schema, and what can be taken from the more extreme narrative psychology positions is that there is a conceptual and structural closeness between schema and narratives that may explain why this is the case. In terms of games, the tendency for us to tell stories may lead us to two hypotheses:

(i) That player will tend to format their gameplay experiences as stories, even when story is not a dominant feature of gameplay. This is analogous to Bartlett's subjects framing the contents of "The War of the Ghosts" to fit their most comfortable means of recall;

(ii) That where the protonarrative units within a game (world, characters, and avatar) can most easily be formed into a network, or where a stronger set of predetermined relationships (the plot) exists, players will find it easier to remember details about these units which are not directly related to gameplay. In other words, by formatting the play experience as a story, nonludically significant detail will be more easily and effectively recalled.

5. THE STUDY

This is all very well in principle and theory, but what empirical evidence can be offered to support it, and how does it relate to the practical business of game development? The following section is a summary of the paper original published at Cyber games 2007, and the reader is referred to this paper for a full analysis of results [1].

A simple study was carried out, whereby twenty-six participants played either Bethsheda's Call of Cthulhu: Dark Corners of the Earth (2006), or id's Doom3: Resurrection of Evil (2005) for 40 minutes in more-or-less natural playing conditions and then undertook a semistructured interview. In this interview, subjects were asked to discuss four factors in their experiences: the world they explored, the characters they met, the avatar they controlled, and the sequence of events—literally "what happened when you played the game?" There were two data sets under investigation. Firstly, the quantity and quality of information about the gameplay experience reported; and secondly, whether there was apparent use of clear story structuring in the reports obtained. Thus, the study aimed to determine whether there was any evidence that a game with a strong and complex story such as Cthulhu (CTH) aided recall or prompted recall with a story-like structure, and whether either of these factors were present or diminished in the relatively unnarrative resurrection of evil (RES). Readers are referred to the original paper for a discussion of the in-game narratives and gameplay of these titles. The key factors of the results are, however, reproduced below.

5.1. Character

CTH subjects demonstrated a generally good grasp of the large cast in the opening levels of the game. When asked "Can you tell me about some of the characters you met in the game?", most subjects responded by talking about classes of characters. Individuals were mentioned less and, interestingly, the most frequently mentioned individuals were not actually met in the game but integrated within goal structure. However, it was intriguing to note that 5 subjects included the avatar in their choices of reported characters.

By contrast, there are only two named NPCs in RES, one (George) is met briefly during the first level and the other (McNeil) being visually introduced in the opening cutscene and represented by occasional radio contact. Alongside, there is an unnamed marine who bequeaths his weapon and immediately dies and various unnamed and doomed voices that occasionally come through over the radio. The opening cutscene shows the player's squad being killed prior to play starting. What is interesting is that given the paucity of NPCs in the game; one might expect the NPCs to stand out; however, only 5 of the 13 players remembered McNeil; none remembered her name correctly. Less than half (5/13) reported George and only one guessed at his name. Half the subjects spoke of the other marines and team mates, suggesting that they "could speak to you," or "they were helping you." Of these, only one noted that the entire squad was wiped out prior to play starting, though it is possible that they attributed the unnamed grabber gun marine to this squad. The CTH subjects were altogether more successful at recalling names. Only 5/13 could not recall a single name from the game, and two of these later remembered.

All subjects from both groups had no problem when asked to provide a motive for one of the characters they had identified. In the CTH group, these were usually fairly accurate, and in many cases picked up on subtle nonludically significant information. Player's asked about the motive for the marines being on Mars in RES which were far less sure and in some cases highly creative in their responses. 9/13 players were asked and the results varied from the semiaccurate "there was an incident," "they used to go there and lost the colony" to the false "they have discovered this archeological site," "conducting some research," and fanciful "human curiosity." Only one subject noted the cover story given in the opening sequence.

5.2. Environment

Subjects were asked to talk about the environments they visited and then prompted with two further questions: any particularly memorable features or details, and what sounds were present? CTH splits into two levels: the opening sequence in a dilapidated cult mansion and its underground tunnels and Innsmouth itself. RES is all set in an archeological dig site, with alien architecture slowly transforming into the human base sited above it. These were variously described as caverns, mines, high-tech industrial, and Aztec. The presence of technology was noted, often (4/13) in relation to the number of boxes and crates lying around. What was most striking about RES subjects descriptions of the environment was how directly indexed to gameplay mechanisms many of them were. 6 of the 13 subjects talked explicitly about generic game devices rather than the presented environment.

The darkness of the levels was the consistent feature noted, with all subjects referencing it. Beyond that, features were evenly distributed between pits, doors, and interactive objects (a power cell transplant sequence was noted by 4 subjects). It was quickly recognised by 5 subjects that each hostile agent was preceded by a signature sound; aside from this, ambient noise was noted. However, no subject reported the radio transmissions that sporadically interrupt the action, nor the direct instructions from McNeil.

The CTH group found it easier to talk about the environments, perhaps due to the diversity of spaces they encountered. 12/13 subjects differentiated between the two playable levels. Two subjects confused the cult house with the asylum in the opening cutscene, which may be attributable to the morgue and experiment rooms in the basement. A further 7/13 used distinct narrative structuring when describing the environments. 4 of the 13 subjects referred to a gameplay mechanism: the save point, the fact that the designers increased tension by reducing the sizes of the environments at critical moments, the reduction of the visual field with movie bars to indicate a cutscene, and the lack of weapons increasing a sense of vulnerability. Finally, 4/13 subjects reported the town's name unprompted (two were correct), which may be interpreted as evidence as it was engaged with on a homodiegetic level rather than just as a game map. It is also worth noting that two subjects questioned the reality of the Innsmouth level altogether, both suggesting that the entire episode was actually a hallucination and that the player had never recovered from the six-year psychotic break that separates the levels. Again,

this may be taken as evidence that they were engaging with the environment at a significantly narrativised level.

5.3. Avatar

One thing both study groups shared was a very distinct conceptual distance between player and avatar. Only two subjects in the entire study referred to the action in the first person. Further, the majority used the second when discussing plot, character, and environment: "you go into the basement," "you are this marine."

However, it is important to note that over identification with the avatar can be problematic, as it exposes the limitations of the game system [23]. The fact that most of the subjects in the study felt that they were controlling Jack or the marine, or in some cases "aiding" them, acting as a team suggests that the avatars were functioning effectively.

Subjects were first asked about their relationships to the avatar, and then whether they thought he had a definable character. If the answer was yes, they were prompted to try and encapsulate this personality in few words. Finally, they were asked about their motives and whether they considered this to be the same as their avatars.

All but one of the CTH subjects easily identified with Jack, citing the amount of background material as the major reason they were able to do so (4/13 also stated that the gameplay device of hearing his heartbeat increase in times of stress helped draw them in). Although they clearly distinguished themselves from him, 6/13 said they felt they were looking through his eyes, or otherwise, operating in tandem with the character. One suggested he felt as if he was playing part of Jack's mind, with the game script providing the counterpoint. All the subjects said they could identify a clear character, and their suggestions of personality fitted those suggested by the game, with stating that the amnesia, whilst giving him a motive to continue playing, was a block to this. The one subject who failed to identify with Jack said that whilst the amnesia gave him a motive to continue playing, it blocked his empathising with the character. Four also inferred personality from his responses to the game's action: "he asks a lot of questions," "he is very curious," "he is not afraid of finding things out," and seamlessly integrating essential gameplay devices with the presented world. Most felt that their motives and Jack's coincided—a drive to find things out, to solve the mystery of not just Innsmouth, but the missing six years of his life.

RES subjects found empathy easy too but struggled more with the notion of character. Although 8/13 felt that the marine had a character, when asked to summarise his personality, there were noticeable pauses; then 5/13 constructed a personality based around their play styles—either "cool, level headed, not freaked out by what is happening" or "a kick-ass marine." The remaining four described the avatar as bland, or a shell, though two of these suggested that as the story progressed, they may understand more about him. One tied his motive to try to find his squad, which is completely missing from the actual game; another candidly pointed out that the initiation of the action comes from the avatar picking up the artifact and that he was playing the "idiot who caused it all." Noticeably, the RES players were more likely (5/13) to differentiate their motive from the avatars: whilst he was variously "trying to get to the surface," "escaping," "staying alive," "returning the artifact for study," they remained only superficially involved, wanting to explore the game, or just responding to wave after wave of hostile avatar. Several (3/13) wanted additional characterisation to flesh out the marine's character.

5.4. Plot

The need for closure was highly evident in both subject groups; most of whom assumed a closed narrative was unfolding, even if they did not fully grasp it. CTH subjects generally coped well with a highly complex narrative, including an unconventional temporal sequence. One subject failed to identify Jack in the opening sequence; another suggested that the suicide was successful and the Innsmouth level was not real. All of the CTH subjects described the plot fully or near fully and did so using clear storytelling structures: there was clear cause and effect and understanding of temporal sequencing. More to the point, every subject thought a story was operating behind the action—two even suggested that it was more important than the action (one describing the experience as more like watching a film than playing a game) and were happy to ascribe the gaps in the information they were given to a plot arc they had yet to uncover although most assumed they would uncover it. Asked if they believed that other characters within the game knew more than they did, all but one answered yes. Certainly, Cthulhu is a mystery game and is quite deliberately aimed at creating this impression. None of the subjects found the action arbitrary—they all assumed that it was only their ignorance of the final plot resolution that hindered their understanding. Conspiracy, and its counterpart, amnesia, is a powerful theme in FPS games, occurring in nearly every title, and it is evident why this should be. Not only does it allow narrative development to be offered as a reward scheme, but it also achieves two more direct gameplay functions. Firstly, it lowers the player/avatar's status, training them to be reliant upon the system for information, which is why it is so often attached to high-status NPCs. Secondly, it allows the system to gain control over information shortfalls: it is simply not necessary to offer a complete package of information if the closure is operating successfully—the player will contribute at least the assumption that all will become clear and, as such, shortfalls and contradictions can be masked.

Tellingly, even though RES subjects struggled to create a full narrative of their experiences, quickly degenerating into brief summaries "monsters come and you shoot them," "you just keep going until to find the boss," and several 3/13 admitted complete ignore as to what was going on; most (7/13) believed there to be a story happening. This would seem to confirm that Kermode's question remains valid in the sphere of game research. Only two drew attention to the PDAs lying around the environment which provided background story. One was convinced that McNeil would be revealed to be the nemesis figure (another essential FPS device). Two subjects noted that the "leaders" (presumably McNeil),

rather than being in possession of extra information, had no idea what was happening or what to do about it; three others correctly recalled her statement that "I have seen this before." Narrative plays a very small part in Resurrection of Evil's action, and there is little in the way of a coherent ecology: demons teleport in according to shock value and challenge, and it is not altogether surprising that 4/13 subjects found the action arbitrary.

6. CONCLUSION

A game with a high emphasis on story such as Cthulhu seems to enable players to recall a substantial quantity of the information it presents, even when this is presented in a nonstandard and incomplete fashion. Although players often fail to remember names, they are adept at either recalling or inferring motive. Even though Cthulhu contains a much higher number of characters than Resurrection of Evil, subjects were able to remember much more about them, suggesting that players of the latter were simply not paying any attention to them. This may sound banal, but it is evidence that the system is training the player to attach significance. Further, the fact that players of Resurrection found it difficult to recall their actions in detail suggests that a strong plot may not only act as a reward scheme but aid in orientation and postexperience affect.

Players in both titles inferred personality from cutscenes and homodiegetic information where this was lacking; they frequently constructed it themselves from the activities of the avatar. The environments and objects of Resurrection without a strong plot structure were recalled often according to gameplay devices, whereas Cthulhu's were placed in a homodiegetic context. Finally, closure was clearly operating across both games, even with a rudimentary narrative; Resurrection players inferred a body of unknown information that many were convinced would be revealed to them, even when they misread the plot—and often remembered little of the primer from the opening cutscene.

Thus, rather than a late bolt-on, in reality no less simplistic, or the alternate reward system, we need to consider story within games as an integrated aspect of the overall system. If utilising narrative devices such as character, plot, closure, and voice has a direct and dynamic impact upon the psychological processes in operation during the act of play. Then, not only does this provide us with a less subjective means of interpreting story in games, but it begins to open up a pseudo-mechanistic approach to putting together effective, ludically functional stories. In order to achieve this, we have proposed that the common issues of narrative can be bypassed if we utilise the concept of protonarratives to envisage story as a network of objects, with a degree of predeterminism in terms of relationships which tend to yield interpretation within a predetermined range.

In a very real sense, this invites us to, one way or another, avoid the "art" of stories in favor of the "craft." In analysing story in games in this way, the mechanics of the application of story to game design can be exposed, making the highly complex field of narratology and its application to psychology accessible to those whose specialisms lie in other areas of game design and development. Further, it highlights the direct impact story can have upon gameplay, in terms of creating more memorable, complex and affective gaming experiences. We are in the position, in terms of the maturation of the both game design and game research, for the medium to demand a wholly distinct and specific understanding of the nature and use of story as a functional technology.

REFERENCES

[1] D. Pinchbeck, "I remember Erebus: memory, story and immersion in first person shooters," in *Proceedings of the 3rd International Conference on Games Research and Development (CyberGames '07)*, pp. 121–129, Manchester Metropolitan University, Manchester, UK, September 2007.

[2] R. Barthes, *Image, Music, Text*, Fontana Press, London, UK, 1993.

[3] D. Carr, *Time, Narrative and History*, Indiana University Press, Bloomington, Ind, USA, 1985.

[4] H. Jenkins, "Game design as narrative architecture," in *First Person: New Media as Story, Performance and Game*, N. Wardrip-Fruin and P. Harrigan, Eds., pp. 118–130, MIT Press, Cambridge, Mass, USA, 2003.

[5] M. Eskelinen, "Towards computer game studies," in *First Person: New Media as Story, Performance and Game*, N. Wardrip-Fruin and P. Harrigan, Eds., pp. 36–45, MIT Press, Cambridge, Mass, USA, 2004.

[6] G. Frasca, "Ludology Meets Narratology: similitudes and differences between (video)games and narrative. Finnish version originally published in Parnasso#3, Helsinki, 1999," English translation, July 2006, http://www.ludology.org/.

[7] F. Bartlett, *Remembering: A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge, UK, 1932.

[8] R.C. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding: An Enquiry into Human Knowledge Structures*, Lawrence Erlbaum Associates, Hillsdale, Mich, USA, 1977.

[9] M. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, P. Winston, Ed., McGraw-Hill, New York, NY, USA, 1975.

[10] F. Kermode, *The Genesis of Secrecy: On the Interpretation of Narrative*, Harvard University Press, Cambridge, Mass, USA, 1979.

[11] D. Pinchbeck, B. Stevens, S. Van Laar, S. Hand, and K. Newman, "Narrative, agency and observational behaviour in a first person shooter environment," in *Proceedings of Narrative AI and Games Symposium: Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISOB '06)*, T. Kovacs and J. A. R. Marshall, Eds., pp. 53–61, SSAISB, Bristol, UK, April 2006.

[12] J. Juul, *Half Real: Video Games between Real Rules and Fictional Worlds*, MIT Press, Cambridge, Mass, USA, 2005.

[13] J. A. Robinson and L. Hawpe, "Narrative thinking as heuristic process," in *Narrative Psychology: The Storied Nature of Human Conduct*, T. R. Sarbin, Ed., pp. 111–125, Praeger, London, UK, 1986.

[14] T. R. Sarbin, "The narrative as a root metaphor for psychology," in *Narrative Psychology: The Storied Nature of Human Conduct*, T. R. Sarbin, Ed., pp. 3–21, Praeger, London, UK, 1986.

[15] E. Tulving, "Episodic and semantic memory," in *Organization of Memory*, E. Tulving and W. Donaldson, Eds., Academic Press, New York, NY, USA, 1972.

[16] M. Eysenck and M. Keane, *Cognitive Psychology: A Student's Handbook*, Psychology Press, Hove, UK, 2000.

[17] M. L. Crossley, *Introducing Narrative Psychology: Self, Trauma and the Construction of Meaning*, Open University Press, Buckingham, UK, 2000.

[18] J. Loomis, "Distal attribution and presence," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 1, pp. 113–119, 1992.

[19] J. Bruner, *Acts of Meaning*, Harvard University Press, London, UK, 1990.

[20] S. Nath, "Narrativity in user action: emotion and temporal configurations of narrative," in *Proceedings of the 4th International Conference on Computational Semiotics for Games and New Media (COSIGN '04)*, Split, Croatia, September 2004.

[21] J. C. Mancuso, "The acquisition and use of narrative grammar structure," in *Narrative Psychology: The Storied Nature of Human Conduct*, T. R. Sarbin, Ed., pp. 91–110, Praeger, London, UK, 1986.

[22] R. C. Schank, *Tell Me a Story: A New Look at Real and Artificial Memory*, Charles Scribner's Sons, New York, NY, USA, 1990.

[23] D. Pinchbeck, "Ludic reality: a construct for analyzing epistemology and meaning-mapping in play," in *Proceedings of the Philosophy of Computer Games Conference*, University of Modena and Reggio Emilia, Italy, January 2007.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 720280, 7 pages
doi:10.1155/2008/720280

Research Article

A Conceptual Framework for the Analysis of First-Person Shooter Audio and its Potential Use for Game Engines

Mark Grimshaw[1] and Gareth Schott[2]

[1] *School of Art and Design, University of Wolverhampton, City Campus, Molineux Street, Wolverhampton WV1 1SB, UK*
[2] *Department of Screen and Media Studies, University of Waikato, Private Bag 3105, Hamilton 3240, New Zealand*

Correspondence should be addressed to Mark Grimshaw, mark.grimshaw@wlv.ac.uk

Received 27 September 2007; Accepted 20 November 2007

Recommended by Kok Wai Wong

We introduce and describe a new conceptual framework for the design and analysis of audio for immersive first-person shooter games, and discuss its potential implications for the development of the audio component of game engines. The framework was created in order to illustrate and acknowledge the direct role of in-game audio in shaping player-player interactions and in creating a sense of immersion in the game world. Furthermore, it is argued that the relationship between player and sound is best conceptualized theoretically as an acoustic ecology. Current game engines are capable of game world spatiality through acoustic shading, but the ideas presented here provide a framework to explore other immersive possibilities for game audio through real-time synthesis.

1. INTRODUCTION

Members of your platoon cluster around you and over the radio comes the message "follow me" with others responding "affirmative." You follow the sound of organ music discovering that it emanates from a Gothic church with buttressed superstructure. In the distance, the sharp crack of gunfire and the dull thud of explosions catch your attention. Eager to join the fray, the metronomic rhythm of your running boots on the hard surface of the path is soon matched by the sound of your panting breath; this quickly overtakes the pace of your slowing footsteps so you slow to a walk. Soon the organ music is left behind and the cacophony of battle intensifies. Suddenly, a siren indicates that a platoon member has managed to steal the enemy's flag and, amidst the sharp staccato of machine gun fire, you see and hear him weaving and running towards you, flag in hand, and hotly pursued by a posse of enemy soldiers.

An account of player experiences when playing first-person shooter (FPS) games is useful in several respects when establishing the groundwork and rationale for the development of the conceptual framework presented in this paper. To begin with, it may intimate a level of engagement with games signaling, among other things, the pleasures and satisfaction at being involved in an individual or team situation. Such pleasures result from (amongst other reasons) a demonstration of individual skill and mastery (in a multiplayer situation, a *public* demonstration of such skill) combined with the pursuit of collaborative objectives in an environment that, in many respects, simulates real-world problem-solving scenarios. The account may also intimate that the perspective presented and the nature of the engagement with the game are both subjective and individually constructed. Indeed, the plot of the very story may be very different when narrated by the other players involved in the same game scenario. Importantly, it outlines how FPS games typically place the player in a hostile environment (the hunter and the hunted) that demands attentiveness to all available cues (especially, and crucially for the aims of this paper, sound cues) for team success, character survival and individual glory. What is described is not fiction or imagination, but a lived account of experiences within the immersive spaces of a particular genre of computer games.

Many of the points raised above may be applied to many computer games (not to mention other forms of gaming) with varying degrees of success and prioritization. Some games provide different cues for the solution of a puzzle, some have less of a team aspect or none at all, others are less combative while others offer different perspectives.

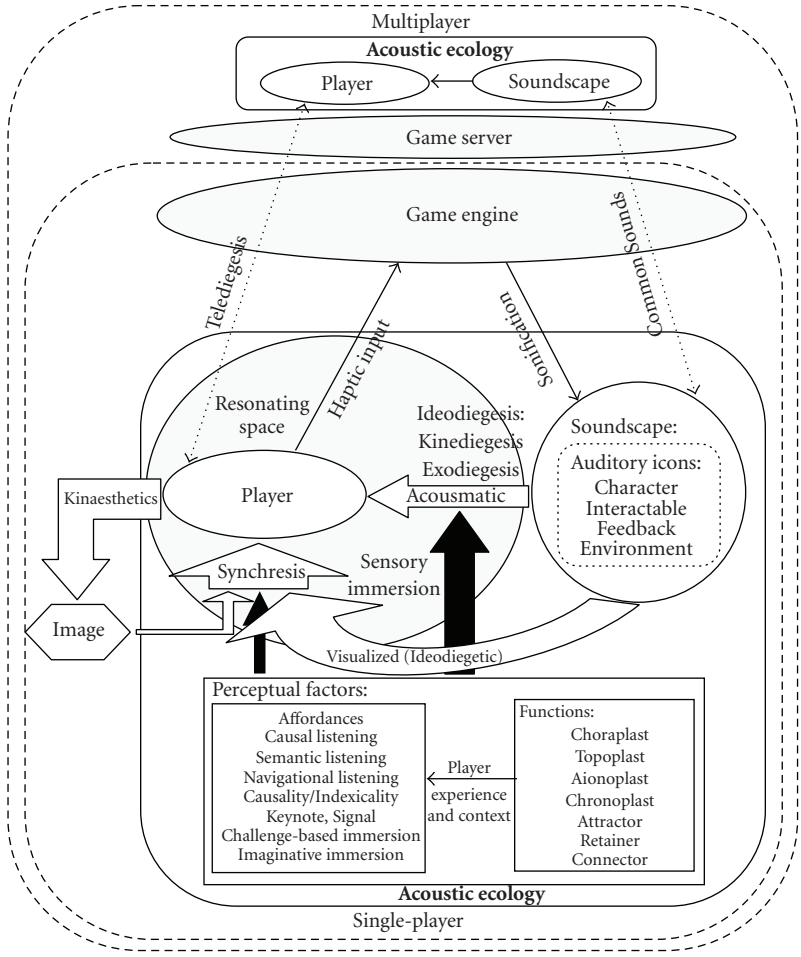


Figure 1: A conceptual framework of the FPS game acoustic ecology in multiplayer mode.

(For a more complex and extensive attempt at taxonomizing computer game types, see *A Multi-Dimensional Typology of Games* [1].) Broadly speaking, it is the types of cues offered, the hostile environment, the mix of team and individual skills, the immersive, first-person perspective, and, of course, the combat that signals the FPS genre. It is our contention that sound cues in FPS games afford more possibilities than in other genres to live the type of game experience signaled in the account. A first-person perspective game uses sound to immerse the player within the game environment in a way that a 2-dimensional platform game such as *Donkey Kong* [2] or a variety of role playing games (RPGs) do not typically attempt. This paper presents a conceptual framework for FPS game audio and a model of game worlds as an acoustic ecology in order to increase the ability of both game scholars and developers to analyze both the relevance of current applications of audio and its effect upon player immersion and player-player interaction.

The conceptual framework (Figure 1) and examples given here account for multiplayer run and gun FPS games, that is, networked games in which there is more than one human player. Bots (computer-generated characters) do not (yet) respond to sound but, in tracking down or evading player-controlled characters, make use of game code variables that change according to that character's position, actions and status. The assumption is made that the conceptual framework for single-player games constitutes a subset of those found in multiplayer games, hence it is the latter that is investigated here.

Of the terminology that resides in the conceptual framework constructed and outlined here, a large proportion is derived from a broad range of disparate work on the nature and function of sound spread across a variety of media. Such areas utilized in the framework include kinaesthetics, affordance theories, modes of listening, auditory icons, diegetic sound, sonification, causality, indexicality, soundscapes and immersion theories to name but a few. In doing so, it was necessary for all these treatments of sound to be adapted to account for the medium through which the FPS genre is experienced, either by pointing out the significant differences (between the medium to which the terminology was originally applied and that of the FPS genre) and adjusting accordingly or by extending the theoretical model to include new terminology where existing terminology proved insufficient. In order to further illustrate the framework, the FPS game *Urban Terror* [3] is used (Figure 2).

Figure 2: A screenshot from *Urban Terror*.

This paper extends a paper [4] presented by the authors at the Cybergames 2007 conference by exploring the implications of the conceptual framework described there for the future development of the audio component of game engines.

2. THE CONCEPTUAL FRAMEWORK

Understanding of sound in the FPS game world is a matter of experience. This experience, and the resultant comprehension, is the result of the training and conditioning which occurs either external to the FPS game (for example, through exposure to popular commercial cinema sound conventions) or which takes place during initial exposure to the sonic conventions of the FPS genre as a whole or to the specific FPS game being played. These conditions apply equally to both the sound designer and the player who, ideally, should have broadly similar socio-cultural experiences and understandings of sound. FPS game sounds may therefore be described as a set of sonic signs or auditory icons which may be analyzed through semiotic terminology, such as indexicality, iconicism, symbolism or metaphor, in an attempt to explain how the intended meaning is (ideally correctly) translated to the received meaning. Thus, the FPS game engine may be understood as a sonification system in which sounds are (re)encodings of non-audio data. This game world data may derive either directly from the game engine, as in the case of game status sounds for instance, or, in the majority of cases, is an expression of player activity, such as the sounds of footsteps or the firing of weapons.

2.1. *Audio sample categorization*

Our first approach in constructing a taxonomy of FPS game sounds was one that perused the classification of audio samples as found on either the distribution medium or on the installation drive of the product itself. This initial approach provided useful insights into the sound designer's classification system which itself may be extrapolated to the meaning that is intended for particular sounds. While this approach has already been used to account for game sound within games studies community through reference to character, interactable or environment sounds, for example, [5–7], none of the literature explicitly examines the distribution or installation media for further clues as to the sound designers' intentions. At the very least, our approach revealed a division between diegetic and nondiegetic sounds as there is typically a separate directory for music or menu interface audio samples as opposed to other audio samples which, themselves, may be subclassified into character, interactable, environment or feedback audio samples. Thus, of the 607 base audio samples in *Urban Terror* (game-specific audio samples as opposed to level-specific audio samples), fully 601 are available to be used during gameplay with the remaining six being the menu music (one) and menu interface sounds (five). The 601 audio samples are, therefore, diegetic whilst the remaining six are nondiegetic.

The game designer-constructed organization of audio samples in *Urban Terror* is illuminating in several respects. Firstly, it is an indication of how the game code deals with sound and its relationship to a variety of characters, objects, and locations within the game. Sounds that players' characters create as they move, fire, or taunt are separated from environment sounds which are part of a location; sounds of interactable objects are separated from the sounds noninteractable objects make, and diegetic sounds are separated from nondiegetic sounds. Secondly, the sheer number of sounds is an indication of the importance of sound to the game experience. Thirdly, this organization of sound indicates some of the technical limitations of the game, namely in the areas of media storage and computer memory. As an example, some audio samples of footsteps are shared between the characters and this decreases the number of sounds which must be stored on the game distribution medium (a compact disc or Internet download in this case) and which must be loaded in the computer memory while playing.

Alone, this mode of categorizing sound offers little insight into the function and meaning of sound, how sound is used in the game by the player or how sound functions to form an acoustic ecology. In order to explore such issues, it was necessary to employ and devise other taxonomic approaches. However, before proceeding to these other possible forms of classification, consideration of the means of sound creation and production at the game design stage is useful as it sheds some light on the degree of interaction made possible in the FPS game which directly relates to the player immersion within and participation in a game-related acoustic ecology. In any computer game, sounds heard during gameplay and from within the game environment are synthesized or digitally recorded and stored as discrete audio samples. In all modern run and gun FPS games, most, if not all such sounds consist of audio samples and this is certainly the case for *Urban Terror*. Most of these audio samples are sounded in response to player input, game status (which, in most cases, is an indication of player activity) or bot activity in games where bots are employed. A smaller number of environment audio samples are under the control of the game engine although their audification may be responsive to player location (by pan and intensity) or where the game engine is capable of acoustic shading.

Such audio samples (as described above) are labeled nomic auditory icons by Gaver [8]. However, in the context of games, *nomic* is an ill-advised term to use, and so we prefer to call them causal auditory icons. They bear a strong degree of causality and indexicality to the actions they represent because they are usually recordings of real-world analogues represented in the game, hence the virtual causality of the sounds. Conversely, a symbolic auditory icon has a more arbitrary mapping between the sound and the event it represents and within games aspiring to a degree of realism (such as *Urban Terror* [9]) there are few such auditory icons.

The abundance of recorded audio samples (as opposed to synthesized audio samples), which may be described as causal auditory icons, combined with their appropriate in-game use (in other words, they are causal sounds with a high degree of virtual indexicality, for example, a recording of a shot-gun is sounded each time a shot-gun is fired), is a good indication of the level of realism the FPS game aspires to. *Urban Terror*, which is usually described as a realism mod, is a prime example; of the 601 diegetic audio samples available, the only synthetic audio samples (the only symbolic auditory icons) are those related to game status events, such as when a flag has been captured. This may be compared to *Quake III Arena* [10] or *Quake 4* [11] which, set as they are in a more fantastical gamescape, have a greater proportion of symbolic auditory icons representing not just game status events but also various audio samples sounded by player input (such as those to do with power-ups and teleporters).

2.2. Diegetic audio

It is possible to classify all audio samples as either diegetic or nondiegetic following film sound theory. However, differences in the creation and resultant nature of the FPS game soundscape compared to the film soundscape require refinements of the term diegetic. As already noted, sounds in an FPS game consist of discrete audio samples and, unlike film, there is no complete game soundtrack that is stored on the distribution medium and played during gameplay. The FPS game soundscape, which forms a part of the acoustic ecology, is created in real-time through the agency of game engine actions (the sounding of game status feedback or ambient audio samples, for example) or through the agency of player input acting upon the discrete audio samples which form the soundscape's palette. Furthermore, with any playing of the game (even the same level), the resultant soundscape will be substantially different for the one player and, in a multiplayer game, the soundscape experienced by one player will also be substantially different to that experienced simultaneously by other players. It is for this latter reason that we define the terms ideodiegetic (those sounds that any one player hears) and telediegetic (those heard and responded to by a player—they are ideodiegetic for that player—but which have consequence for another player; they are telediegetic for the second player). Furthermore, ideodiegetic sounds may be classified as kinediegetic (sounds initiated directly by that player's actions) and exodiegetic (all other ideodiegetic sounds).

Of the class of diegetic audio samples, and in the context of a multiplayer game, all global feedback sounds (such as game status messages) may be classed as exodiegetic sounds. They are ideodiegetic in that they are heard by all players (simultaneously) but initiated by the game engine in response to significant events. All other audio samples may be ideodiegetic or telediegetic depending upon context. These include environment sounds (which are usually level-specific audio samples rather than game-specific audio samples). Ideodiegetic sounds may be classed as either kinediegetic or exodiegetic. For the player who triggers them, the sounds are kinediegetic; they are exodiegetic for other players. If such sounds have consequences for other players who do not hear them (for example, the blast of the shotgun which kills an enemy may draw others of her teammates to that location which itself may provide opportunities for the opposing team), they may also be classed as telediegetic for these other players.

As has been noted by several writers [6, 7], sound in an FPS game may be attended to in one of three modes: reduced listening, semantic listening, and causal listening. Reduced listening, as noted by Stockburger [7], is little used by experienced players. What these writers do not suggest is that the mode of listening may change depending upon context and experience. Furthermore, we identify a fourth mode, navigational listening. This is required because of the unique (compared to electro-acoustic music and film sound theory where the original three modes were first described) abilities of the FPS player to move her character around the 3-dimensional game world. In this mode, certain sounds may be used as audio beacons helping to guide players, especially those new to the particular game level, around the game world structures.

2.3. The FPS game soundscape

Schafer's [12] keynote sounds are, in this context, audio samples which form part of the sonic ambience and may not be directly triggered by the player being, instead, sounded by the game engine. (They may be triggered by other players but are judged by the one player to be distant and of little interest and so form part of the general ambience of battle.) An example in *Urban Terror* is the Bach organ fugue in the *Abbey* level or, in the same level but in a different location, the twittering of birds. There is some ambiguity here that is not captured by the brief descriptions of such environment or ambient sounds in existing games studies literature. For example, the player does typically have some kinaesthetic control over the sounding of these sounds; by simply moving away from a location, the sound may be attenuated to silence (and vice versa). Furthermore, if a keynote sound is a sound which is not intended to be consciously listened to, merely forming the background for more perceptually important sounds, the decision to consciously attend to a sound or not is often a matter of player choice, indeed, musicians may respond to the organ fugue differently to non-musicians.

A signal sound is a foregrounded sound which is designed to be consciously attended to because it potentially contains important information encoded within it. Most of the game-specific audio samples in *Urban Terror* may be classified as signal sounds when they are sounded in a context which foregrounds them. Thus, the loud, and therefore

proximate, sounding of gunshot samples is worthwhile paying attention to (particularly in the individual deathmatch game mode). However, if the sounds of battle are distant, they may be classed as keynote sounds particularly if they are relatively constant and the player's attention is directed elsewhere. All game status indicators and team radio messages are signal sounds because, although they are as pervasive as keynote sounds, they are usually louder and therefore more proximate and, in the case of radio messages, have no reverberation, thereby foregrounding them through the lack of depth cues.

Soundmarks are identifying aural features of the acoustic ecology and may be either signal sounds or keynote sounds which are consciously attended to. Symbolic auditory icons, such as flag status signals, are more likely to be uniquely identifying of an FPS game than causal auditory icons because the latter are derived from recordings of existing real-world, external sounds whereas the former reference the internal game world.

2.4. Immersion through sound

FPS game sounds may be categorized according to a variety of immersive principles. Following Ermi and Mäyrä's ideas [13], all FPS sounds can contribute to sensory immersion where the sounds of the game world override those in the player environment. It should be noted, though, that the degree of sensory immersion is dependent upon a range of factors beyond the control of the game designers including the relative loudness of the two sets of sound and the audio hardware used; one of the factors influencing the decision of most FPS players to use headphones [14] is likely to be a greater sensory immersion. Many sounds offer challenge-based immersion by requiring a response which includes the use of both mental and kinetic skills. It is typically the case that these sounds are ones which are produced by other players and they usually relate to actions involving weapons. However, whilst most level-specific environment sounds in *Urban Terror*, for example, generally do not offer challenge-based immersion possibilities, audio beacons require the navigational listening mode and, therefore, mental skills.

Sounds offering imaginative immersion possibilities are those which help the player identify with her character and the game environment and action. In the first case, FPS games offer a range of character sounds, some of which may be classed as proprioceptive sounds (such as the character breathing whose rate may vary according to the exertions of the character) and which, with a high level of immersion, may be seen as aural prostheses similar to the prosthetic limbs seen receding into the screen. Exteroceptive sounds affording imaginative immersion through identification include a range of sounds which aid in contextualizing the player character within the environment.

McMahan categorises computer game elements as perceptual sureties, surprises, or shocks [15]. The latter militate against immersion in the game world by being external stimuli (or errors in the game) that remind the player that this is just a game taking place within the player's real-world environment. Sureties are mundane cues, expected details providing an experience which is consistent with the rules and conventions of the game world. The creaking of a door as it opens and closes or the footsteps of a player moving around are aural examples of this. Surprises, according to McMahan, consist of three types: attractors (inviting the player to do something); connectors (helping player orientation) and retainers (causing the player to linger in game world locations).

A variety of sounds in FPS games fulfil these requirements. Indeed, any sound inviting an active response may be said to be an attractor. Thus, the sound of gunfire in the distance may tempt the player to investigate and team radio messages detailing enemy actions invite a response on the part of the team player. Many sounds, particularly environment sounds, function as connectors and they are often attended to in the navigational listening mode. Locational and depth properties are important parameters of sounds functioning as connectors. Although, at first playing of the game, the player may derive enjoyment out of certain sounds and so may linger in a particular location in order to hear more, the nature of the FPS game is such that more-or-less continual movement is usually required of the player to seek out or avoid the enemy or to attack the enemy base and so, for the experienced player, no sounds in the FPS game may be said to be retainers.

We propose four terms to describe the spatializing and temporal affordances offered by FPS game sounds. The perception of a variety of spaces is one of the main contributing factors of FPS sounds to the perception of, and immersion within, the game world. In terms of our phrase resonating space, there is a real resonating space, which is the acoustic volume enveloping and morphing around the player, and a virtual resonating space, matching, through a process of synchresis, the illusory visual space depicted on the screen (other virtual spaces may be identified as separate volumes within the game world), the perception of which is created by parameters of sound such as localization, depth cues and reverberation. Such cues may be processed in real time (acoustic shading) with more sophisticated game audio engines or they may be encoded into the audio samples on the distribution medium. Sounds providing this affordance are choraplasts. Sounds may also function as topoplasts where they create the perception of paraspaces such as locations in the game. Additionally, sound may provide the affordance of the perception of time passing or of a particular temporal period in the past, present or future. The former are chronoplasts and, because sound is vectored through time, that is, it takes time to hear a sound, all sounds have a basic chronoplastic function (in addition to any explicit function they may have in this area). The latter are aionoplasts and weapon sounds typically have this function setting the game, for example, in the modern era rather than the mediæval age.

3. THE IMPLICATIONS FOR GAME ENGINE DESIGN

As previously stated, modern FPS games make use of audio samples that may be treated as causal auditory icons (the most common form and typically recordings of real-world events) or symbolic auditory icons (more common in games

with a less realist scenario). Whilst the use of causal auditory icons provides quite accurate sonic representations of real-world artifacts in the game world, that use comes at a price that is calculated in memory, both distribution medium storage and game system random access memory (RAM). Assuming a 16 bit, monophonic, 44.1 kHz game audio system, 100 one-second audio samples require a total of almost 9 MB of storage.

This may not seem expensive with the technology available in 2007. However, there are several factors conspiring to push this cost up. Firstly, many game audio samples are longer than one second particularly if they are vocal audio samples. Secondly, and perhaps more importantly, the game designers' desire to provide the player with an increasingly rich sonic environment requires, as an initial solution, the provision of yet more samples. Unlike sound design for a film, game sound designers work to a non-linear script and "it is not possible to make every gunshot sound unique if you do not know how many gunshot sounds are needed!" [16]. Games such as *Urban Terror* (based on the *Quake III Arena* game engine) must therefore strike a balance between a wish to provide an audio sample for every sonic possibility (an infinite number) and paying regard to storage and memory requirements (*Urban Terror* is typically provided as an Internet download).

Later game engines, such as that used by *Half-Life 2* [17], use acoustic shading techniques. This provides a part solution to the audio sample memory problem by real-time processing of audio samples with reverberation that approximates the virtual acoustic properties of the character's environment. This is also a step towards solving the non-linearity enigma as expressed by Boyd because any one audio sample does multiple duty by having different reverberant characteristics in different locations of the game. However, acoustic shading of audio samples still requires a high use of memory and it provides solely a reverberation solution without taking account of other encodings possible in sound (its emotive aspects or the direct sound source, for example).

Gaver notes that, using what he calls *everyday listening*, sounds are usually described by one or two of their salient characteristics (object and action); a metallic clang, a wooden thump, a glass-like shattering, for example [18]. Indeed, he provides algorithms to show it is possible to conceptualize and synthesize sound according to these characteristics rather than directly by the use of properties such as frequency and intensity; a top-down model as opposed to a bottom-up model. The resultant caricature sounds should then provide the minimum information required to enable at least an approximate identification of both source object and action.

Populating an FPS game with such caricature, synthesized sounds would seem to militate against realism and the requirements of a rich, immersive player experience. However, there is evidence to suggest that, where sound is concerned, a reduced realism may be all that is required to achieve the desired immersive effect [19–21]; a simulation, rather than emulation, that is based upon convention, expectation, and caricature. Certainly, in film, this is how many sound FX work; recordings are made (and enhanced in post-production) of sources and actions that are not necessarily the same as those depicted on the screen. However, by matching the main characteristics of the recorded sound to the expected sound of the screen depiction (such expectation often being the result of cinematic convention) and synchronizing sound and visual action, the audience is persuaded that that screen event really has produced that sound.

The suggestion, then, is that real-time synthesis of sound in digital games may prove to be of benefit in dealing with the twin challenges of memory and non-linear practice without imperiling the immersive experience and, indeed, perhaps enriching it. The ideal scenario might be to have a combination of synthesis and audio samples because (despite advancements in synthesis) some sounds (such as the human voice) are still best represented by audio samples. Synthesis may be of use for the more symbolic auditory icons, fast-repeating sounds (such as gunfire), and background, keynote sounds and may be enhanced through any acoustic shading the game engine offers.

4. CONCLUSION

All sounds, or the use of some sounds, in the FPS game contribute in some way to player immersion in the acoustic ecology and it is this immersion within (and the player's creative participation in the game's acoustic ecology) that, in large part, affords the perception of immersion in the FPS game world. Thus, the player is physically immersed in the real resonating space and, through kinaesthetic techniques and the ability to trigger a range of sounds through various input methods, is drawn into the virtual resonating space that is then synchretically mapped to the visual game world and activity that are represented either on- or off-screen.

The model in Figure 1 exhibits all the elements of the conceptual framework described thus far. Because it is a model of an *acoustic ecology*, it importantly shows relationships between the player (the listener) and soundscape. As it is a model of the acoustic ecology of the FPS run and gun game, though, it includes a variety of components and relationships which are unique to digital games (some of which may be unique to the subgenre) such as the game engine, image, a range of spatial and immersive elements and perceptual factors. Furthermore, because this is a model of a multiplayer game acoustic ecology, it also includes the game server and other players and their soundscapes. (For clarity, only one other player and soundscape are shown here.)

Although the conceptual framework and the model are focused on FPS games as their paradigm, it may well be the case that they (or aspects of them) may also be used to analyze the wider area of digital game sound in the future with the caveat that much further research and testing (using different digital game genres) is required. Furthermore, it is suggested that the conceptual framework and model may prove to be of use in the design of the audio component of game engines by supporting the notion that real-time synthesis of sound in the game is a valid means of providing an immersive acoustic ecology.

REFERENCES

[1] E. Aarseth, S. M. Smedstad, and L. Sunnanå, "A multi-dimensional typology of games," in *Proceedings of the Digital Games Research Conference*, pp. 48–53, University of Utrecht, The Netherlands, November 2003.

[2] "Donkey Kong [Computer program]," Nintendo, 1981.

[3] Silicon Ice, "Urban Terror [Computer program]," Version 3.7, 2005.

[4] M. Grimshaw and G. Schott, "A conceptual framework for the design and analysis of first-person shooter audio," in *Proceedings of the 3rd International Conference on Games Research and Development*, Manchester, UK, September 2007.

[5] Xamot, "Urban Terror," August 2001, http://everything2.com/index.pl?node_id=1036283.

[6] J. Friberg and D. Gärdenfors, "Audio games: new perspectives on game audio," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, Singapore, June 2004.

[7] A. Stockburger, "The game environment from an auditive perspective," in *Proceedings of the Digital Games Research Conference*, University of Utrecht, The Netherlands, November 2003.

[8] W. W. Gaver, "Auditory icons: using sound in computer interfaces," *Human-Computer Interaction*, vol. 2, no. 2, pp. 167–177, 1986.

[9] C. Law, "Urban Terror!," GameSpy, June 2006, http://archive.gamespy.com/legacy/spotlights/urbanterror_a.shtm.

[10] id Software, "Quake III Arena [Computer program]," Activision, 1999.

[11] id Software, "Quake 4 [Computer program]," Activision, 2005.

[12] R. M. Schafer, *The Soundscape: Our Sonic Environment and the Tuning of the World*, Destiny Books, Rochester, Vt, USA, 1994.

[13] L. Ermi and F. Mäyrä, "Fundamental components of the gameplay experience: analysing immersion," in *Proceedings of DiGRA 2005 Conference Changing Views—Worlds in Play*, Toronto, Canada, June 2005.

[14] S. Morris, "First person shooters—a game apparatus," in *Screenplay: Cinema/Videogames/Interfaces*, G. King and T. Krzywinska, Eds., pp. 81–97, Wallflower Press, London, UK, 2002.

[15] A. McMahan, "Immersion, engagement, and presence: a new method for analyzing 3-D video games," in *The Video Game Theory Reader*, M. J. P. Wolf and B. Perron, Eds., pp. 67–87, Routledge, New York, NY, USA, 2003.

[16] A. Boyd, "When Worlds Collide: Sound and Music in Films and Games, 2003," Gamasutra, September 2004, http://www.gamasutra.com/features/20030204/boyd_01.shtml.

[17] Valve Software, "Half-Life 2 [Computer program]," Electronic Arts, 2004.

[18] W. W. Gaver, "How do we hear in the world? Explorations in ecological acoustics," *Ecological Psychology*, vol. 5, no. 4, pp. 285–313, 1993.

[19] B. Laurel, *Computers as Theatre*, Addison-Wesley, New York, NY, USA, 1993.

[20] M. Back and D. Des, "Micro-Narratives in Sound Design: Context and Caricature in Waveform Manipulation, 1996," December 2007, http://www.icad.org/websiteV2.0/Conferences/ICAD96/proc96/back5.htm.

[21] C. Fencott, "Presence and the Content of Virtual Environments, 1999," August 2005, http://web.onyxnet.co.uk/Fencott-onyxnet.co.uk/pres99/pres99.htm.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 176056, 7 pages
doi:10.1155/2008/176056

Research Article

Ambient Games, Revealing a Route to a World Where Work is Play?

Mark Eyles and Roger Eglin

Advanced Games Research Group, School of Creative Technologies, Universtiy of Portsmouth, Eldon Building, Winston Churchill Avenue, Portsmouth, PO1 2DJ, UK

Correspondence should be addressed to Mark Eyles, mark.eyles@port.ac.uk

Received 27 September 2007; Accepted 7 January 2008

Recommended by Kok Wai Wong

A novel way of playing games called ambient gaming is defined and described. Growing out of ideas in ambient music, ambient gaming is defined as "ignorable as it is interesting" after Brian Eno's description of ambient music. Ambient gaming is set in the context of existing games. Further, ambient games are set in a technological context, showing that the technology enabling their development is now becoming available. The specification and implementation of an ambient game prototype, Ambient Quest, are described. Finally, future directions leading to work enhancing games are suggested.

1. INTRODUCTION

This article seeks to address four questions around the concept of *ambient games*. What might the game equivalent of ambient music play like? What technologies would this require? How could an ambient game be produced (with limited resources)? What is the potential for exploiting ambient games? The answers to these questions build on work outlined in previous conference papers on ambient games in which they were combined with role playing games [1] and in which a pilot study of an ambient game simulation was described [2]. Practical suggestions are offered for building and running simple ambient games, and for future ways of exploiting this technology to turn work into play.

2. AMBIENT MUSIC

Brian Eno coined the term *ambient music* on his album Ambient 1: Music for Airports released in 1978. In the sleeve notes of Music for Airports, Brian Eno gives a definition of ambient music, *ambient music must be able to accommodate many levels of listening attention without enforcing one in particular; it must be as ignorable as it is interesting.* In a talk given for the long now foundation's series of seminars about long-term thinking in 2003 [3], Eno talked about Music for Airports: "I wanted to make a kind of music that would actually reduce your focus on this particular moment in time that you happened to be in and make you settle into time a little bit better" [4].

The description of ambient music and the ambient pieces produced by Brian Eno serves as a guide to the creation of an *ambient games* definition and acts as a useful reference point and context for the creation of ambient games.

3. DEFINING COMPUTER GAMES

A video or computer game is an interactive entertainment played against, or with the aid of, computer-generated characters or tokens in a computer-generated environment. A single player game has a series of interesting obstacles to overcome in order to gain rewards. A multiplayer game has a series of interesting obstacles to overcome at the expense and/or with the help of other players to gain rewards. Games require a commitment of time and effort from the player. This commitment varies widely between games. A large and involving strategy computer game like Civilization 2 (or any other game in the Civilization franchise) may require many hours of play and thought from the player. There is also a substantial learning curve at the start of the game before the player is able to play proficiently. When starting a game like

this, the player is often committing themselves to 50+ hours of play, stretching over weeks or months. A simple game like Tetris has a very shallow learning curve (the player can start playing almost immediately) and requires very little commitment from the player, though players may choose to make a larger commitment of time and effort to the game if they wish. The word completion game hangman requires very little commitment and has virtually no learning curve for the average person.

Another characteristic of games is where they are played. For example, console games are played in a single location, the console does not move around during play, though the player may move a small amount while playing. A game on a mobile phone does not normally require the player to move around while they are playing, but does allow the player to move around if they wish to, though there are a small number of games that do require movement. Mobile phone games may be played in any location (which has conditions that will not damage the phone). The game may be played in any location (which has conditions that will not damage the phone). Similarly, the class of games labelled *pervasive/ambient games* allows the player to move freely around everyday locations while playing. The player can play ambient games in the environment they normally inhabit. There are also games that require the player to move around while playing, especially outside of computer games, frequently in locations specially prepared for the games (such as football fields, tennis courts). There are many sports in which the players are required to move around. The rules for cross country running require the participants to move a large distance.

By plotting player commitment against the distance the player may travel while playing, it becomes clear that there is an undefined class of games that do not require large commitments and in which the player may move around (perhaps be required to move around) while playing. This class of games has been labelled *pervasive/ambient games*. See the *Commitment and movement when playing games* figure. The areas marked out for games in this figure are not definitive, there are plenty of exceptions, but they are intended to broadly indicate general areas of commitment and movement.

This figure has a threshold marked showing the commitment required to start playing any game. Below this threshold, the player has not yet committed to starting playing. At, and above, this threshold, the player is deciding to start, and is actually starting, a game. Below this threshold, they have not got the will, or intention, to start playing. However, they may be starting to collect data that will be used in play before they make this commitment.

As implied by the *Commitment and movement when playing games* figure, the key component of an ambient game is that the player may choose their level of interaction with the game.

4. SPECIFYING AN AMBIENT GAME

An example of a game that requires minimal player intervention is Progress Quest (http://www.progressquest.com) [5].

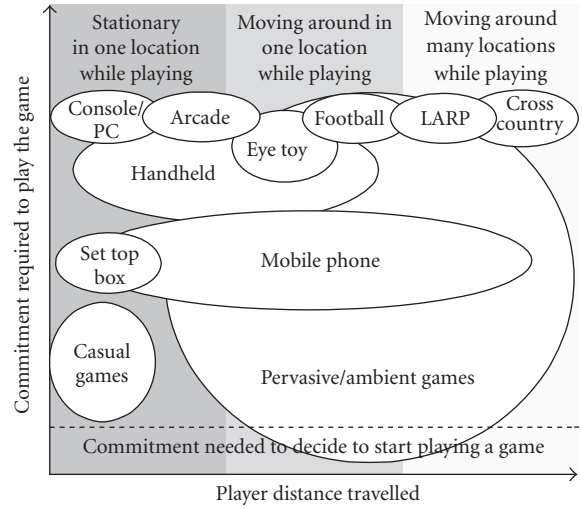


Figure 1: Commitment and movement when playing games.

All a player needs to do in order to play Progress Quest is to set the game running. The game displays the player's character's role playing statistics and lists the completed quests and so on. The player need not intervene again as Progress Quest continues to play, all gameplay decisions are made automatically.

In order to play Progress Quest, the player has to start the game running, consequently it may be argued that there is some (minimal) participation from the player. However, it is not possible to play the game without starting it running. A player might ask someone else to start the game running for them, but they would still have made a decision to play the game, they would be actively involved. There is no possibility of playing the game unknowingly. Even if the players were to set up a program that started Progress Quest at random times while the computer was turned on, they would still have made a decision to start the game.

Progress Quest shares characteristics with films. The game requires the same sort of interaction that is necessary for watching a conventional film on DVD. The player starts the DVD playing and then watches the story unfold on a screen. The difference between a film and Progress Quest is that in a film all the decisions have already been made. The viewer will see the same sequence of scenes whenever they watch the film. Progress Quest generates the story in real time. Players assume a new identity every time they start a new game of Progress Quest and become participants in a new story.

Imagine a game similar to Progress Quest in which after the game starts, the player's actions in the real world affect progress in the game world. The game world consists of a virtual environment containing quests to complete (achieved by defeating monsters at various locations). In this game, the player chooses the degree to which they wish to manage events in the game. At one extreme, the game runs itself, gathering data from the player's actions in the real world and automatically applying this to the game world. At the other extreme, the player can determine how the real world data

is applied in the game world, micromanaging game interactions.

The player may choose to manipulate their actions in the real world to generate data required to progress more successfully in the game. Alternatively, they may ignore the consequences of their actions in the real world and allow data to gather without consciously changing their behaviour. The player experiences a game-induced mood while playing. This is an ambient game.

An ambient game would normally be expected to contain goals for the player, though it would be possible to implement a very open-ended ambient game in which there are no clearly defined goals with the player taking part in much freer ambient play, (akin to Caillois' paidia [6]). However, even in this ambient play the player is still operating within a set of game rules, and might set their own goals within that framework.

An ambient game is coincident with real life; elements are superimposed on the real world. In an ambient game, the gameplay is in the background, available for the players to focus their attention on it. In other types of games, the player has information *pushed* at them, they are required to interact. In an ambient game, the players *pull* information from the game when they want it. Ambient games feature pull, not push, technology. Compare this to ambient music which is composed to be in the background, though the listener can bring it to the foreground and focus their attention on it if they wish.

At the heart of ambient games is the idea that the players can dip in and out of the game; that the game is running in the background, creating as mood, while they are engaged in other activities.

5. AMBIENT GAME TECHNOLOGY

There are two possible ways that an ambient game could be implemented; either the player carries an ambient gaming system around with them or the game is embedded in the environment that surrounds the player. For a truly ambient game, the interface should be unobtrusive, allowing the player to easily switch between engaging and ignoring it. An ambient intelligent environment offers an ideal solution for the implementation of ambient games. There are a number of technologies and ideas that make the production of an ambient game possible.

There are a number of different technologies that are enabling the development of ambient intelligence: interconnectivity, artificial intelligence, and the proliferation of computers. These technologies support the ubiquity, transparency, and intelligence of ambient intelligence [7].

Ubiquity refers to ubiquitous computing [8] in which a massive number of interconnected computers are embedded in the environment.

Transparency indicates that ambient intelligence environments are invisible and in the background [7].

Intelligence relates to the interfaces and ways these interconnected computers respond and interact with people through user friendly interfaces. They are able to *exhibit specific forms of social interaction* (ibid).

The European Union's Information Society Technologies Advisory Group (ISTAG) predicts that ambient intelligence will emerge from the convergence of the following three key technologies:

(i) ubiquitous computing,
(ii) ubiquitous communication,
(iii) intelligent user-friendly interfaces [9].

Ambient intelligence systems may also require locative information, specifying player location and also assigned identity and/or personal identity knowledge, they may need to differentiate between different people. For example, if one of the functions of an ambient intelligence is to control the lighting within a house, it not only needs to be able to turn lights on and off as people move through the house, but also to set brightness levels according to the preferences of individuals.

In order to fulfil the transparency requirement communication with ambient intelligences should be seamlessly integrated into the environment. Computer workstations or input panels do not fulfil transparency. The user might expect to be able to communicate with ambient intelligences through speech or gestures, with the ambient intelligences responding in speech or with their available interfaces (perhaps momentarily dimming lights to indicate that a request has been received and stored).

As devices proliferate, it becomes useful to be able to identify them. Different components in ubiquitous systems need identity in the same way that in games each of the nonplayer and player characters need identity. If there was no way of identifying individual components then it would be impossible to know the outcome of interactions.

Items may be tagged in the physical world with radio-frequency identification tags (RFID). These are transponders that respond with a unique serial number when a reader sends a signal to them. They are frequently used for tracking goods through supply chains, where it is useful to know the location and identity of the goods [10].

People may be tracked in the real world using face recognition systems [11]. This recognition has great implications for ambient intelligence environments where they might be used to recognize and track people as they move around and also to ensure that the systems respond appropriately to known individuals [12].

With the emergence of ambient intelligence and ambient intelligence environments, the age of "massively many *intelligent* computers—one user" is arriving and bringing with it new gameplay opportunities.

6. AUGMENTED REALITY AND PERVASIVE GAMING

The introduction of existing, commercially available, devices able to track the position of people in the world, such as global positioning systems, is opening up many opportunities to explore new ways of playing games, for example augmented reality and pervasive gaming.

Players of augmented reality games wear a head mounted display that allows them to move around the real world with computer-generated images superimposed onto the real

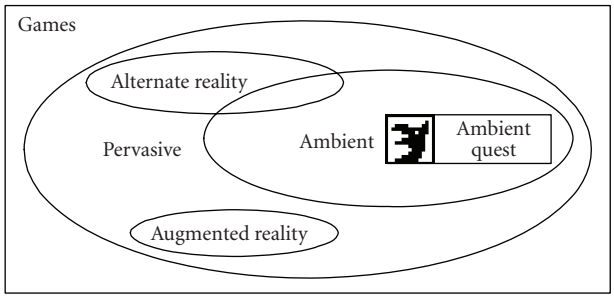


Figure 2: Pervasive and ambient games.

world. The equipment tracks the player's location and where they are looking then generates appropriate images.

Examples of augmented reality games include Pac Man at the Mixed Reality Lab, Nanyang Technological University [13] and ARQuake at the University of South Australia [14].

Augmented reality technology gives one route forward for gaming in the environment, but still requires the users to carry equipment around with them. In its current incarnations, augmented reality requires the user to make a substantial commitment of time and effort to play the game and consequently does not sit within ambient games.

Other pervasive games also feature locative technology so players (though not necessarily all of the players) move into the real world while playing and their position and actions in the real world affect, and are affected by, events in a virtual world [15]. There are a number of different variations on this; the IPerG research consortium lists the following areas of pervasive gaming that they are exploring:

(i) crossmedia games,
(ii) socially adaptable games,
(iii) massively multiplayer reaching out,
(iv) enhanced reality live role-playing,
(v) city as theatre (ibid.).

These vary from ambient games in the commitment the player makes to the game and the intention of the game; the intention of ambient games is to create a mood in an environment. The games listed above also frequently require the player to carry around hardware for playing the game.

The mobile game Feeding Yoshi was presented in 2006 as a game that exploited *seamful design*. Seamful design incorporates discontinuities (gaps and edges) in ubiquitous systems (e.g., wireless and global positioning coverage) in applications, including games. In Feeding Yoshi, the players (working in teams) each carry a PDA as they go about their normal lives in applications, including games [16]. When they are within range of a *plantation* or Yoshi, the PDA beeps and displays details of the plantation/Yoshi. The player can then plant seeds, harvest fruit in the plantation, or feed the virtual Yoshi character. The plantations and Yoshis are generated within the PDAs based on the type of wireless networks detected—secure networks become Yoshis, open networks become plantations. Players can further swap seeds and fruits with each other [17]. Feeding Yoshi is an interesting example of a game in which players play while going about their normal lives. However in order to play the game, the players have to focus their attention onto the implementation of the game on the PDA where they carry out particular game moves; further the game calls their attention, pushing them into the game to make moves when appropriate networks are detected. Although players have a choice of whether to play or not, this game does not share the ambient ignorability of ambient games, in which play is possible without focusing attention on the game.

In 2005, *ambient utility games* were proposed by Kangas et al. in which playfulness is introduced into experiencing and understanding information, especially using innovative interfaces to allow player movements and so on to feed into the game [18]. The purpose of these games is not only to entertain but also to have a utility, or usefulness, to them, encouraging players to perhaps exercise or study: "Utility games belong to a new type of games *where utility is emphasized in the content*" (ibid). The emphasis on utility sets these apart from the ambient games such as ambient quest where the primary intention is to create a mood, not to create useful behaviour in the player; though as a byproduct of playing a game like Ambient Quest, the player may engage in useful behaviour. In the conclusion to their paper, Sonja and Outi do specifically mention leisure gaming as an application of their ambient utility games, though this does seem to contradict the emphasis they place on usefulness and education elsewhere in the paper.

Alternate reality games (ARG), sometimes known cross media entertainment (XME), are, according to an article on CNET http://www.news.com, "... an obsession-inspiring genre that blends real-life treasure hunting, interactive storytelling, video games, and online community ..." [19].

The Alternate Reality Gaming Network defines alternate reality games as "an intensely complicated series of puzzles involving coded web sites, real-world clues like the newspaper advertisements, phone calls in the middle of the night from game characters, and more. These games (which are usually free to play) often have a specific goal of not only involving the player with the story and/or fictional characters but also of connecting them to the real world and to each other. Many game puzzles can be solved only by the collaborative efforts of multiple players, sometimes requiring one or more players to get up from their computers to go outside to find clues or other assets planted in the real world." Unlike augmented reality games, they do not normally require special equipment to be carried around by the players while they are being played, though players are likely to need access to computers, phones, and other sources of information [20].

Many alternate reality games are used for promotional purposes, for example the first alternate reality game, The Beast, was used to promote the film AI: Artificial Intelligence in 2001. More recently, 2006, Volvo cars has used alternate reality game *The Hunt* to promote the release of a new XC90 car (http://thehunt.volvocars.net/uk/thehunt). The alternate reality game Perplex City (http://www.perplexcity.com) is not a promotional tool, but makes money from selling clue cards to players. As well as clue cards, Perplex City also delivers puzzles and clues via websites, podcasts, emails, texts, and live events.

Alternate reality games combine events in virtual computer spaces and the real world to create a coherent gaming experience. They are frequently multiplayer, requiring co-operation between two or more players to solve puzzles and progress. The unfolding stories in these games blur the boundaries between reality and fantasy by incorporating game elements into the real the world that influence game play in online worlds.

Alternate reality games are very close to ambient games, but still require a commitment from the player and demand specific game playing behaviours. They are not primarily driven by normal everyday behaviours.

7. DEFINING AMBIENT GAMING

Ambient gaming has been specified in relation to ambient music and the technologies that would make ambient gaming possible, in particular the development and nature of ambient intelligence environments has been described. Although ambient games are likely to include computer game technology to create their virtual worlds they are not the same as traditional computer games. New ways of playing computer games in the real world, such as augmented reality gaming, pervasive gaming, and alternate reality games, have been described in order to set the idea of ambient games in context.

As previously stated, ambient games can be defined as games that are controlled by everyday actions (i.e., not using a dedicated game input device, mouse or keyboard) in everyday, real world environments that have gameplay consequences in a virtual game world. Further, ambient games do not demand the attention of the player, they are "ignorable as they are interesting" [3], allowing players a wide depth of interventions from letting the game play itself to micromanaging game events. Ambient games are always *on*, the player does not experience them in isolated, discrete playing sessions as is the case with, for example, console games. Ambient games also allow the player experiences that range from superficially shallow to profoundly deep. The player is able to choose how they focus their attention on the game, and alter their degree of attention at will. A key attribute of ambient games is their intention and ability to create a mood in an environment.

Ambient games are coexistent with the real world and may be seamlessly controlled by the intelligent interfaces of ambient intelligent environments. They are intended to influence the player's experience of their environment, perhaps invoking emotions through the game that affect the player's perception of the real world. The ambient intelligence interfaces give information (feedback) on the progress of player characters (avatars), and allow the player to interact with the game's virtual world through gesture, speech, and movement. In some applications of this game technology, the player's heart rate, respiration, and so on might also be used to control their avatar, rather like Dan Sutch's Fizzees (Tamagotchi-like digital creatures that are nurtured by the physical actions of their owner) [21]. In other applications it may be things that players do in the real world such as moving around, and spending money. The feedback to the player may not necessarily be limited to auditory and visual senses, but might also affect other senses, perhaps creating temperatures, smells, and so on.

The ambient game definition allows for single player, multiplayer, or massively multiplayer gaming. The number of players does not affect the *ambience* of ambient games, and they may often feature social interactions between players.

8. IMPLEMENTING AND RUNNING AN AMBIENT GAME SIMULATION

The game Ambient Quest is an example of a simple, inexpensive, single player ambient game simulation. In order to test out some of the ideas of ambient games with a moderate budget, Ambient Quest has been designed to be played using a simple 2D virtual world, and the ambient intelligent environment has been somewhat simplified, being simulated by carrying a pedometer to measure distance travelled. In the current implementation of Ambient Quest, the number of steps taken by the players is used to determine distance travelled in the virtual game world. In the first version of Ambient Quest, Players entered these distances into the game by giving them to the researcher (in person or via email) who takes on the role of the intelligent interface and enters the distances manually into a game engine that then provides a log file that may be used by the players to display the activities of their avatars. A more up-to-date version of Ambient Quest (version 2.1) allows players to directly enter their own distances.

The limitations of this system are that the players have to make a little more commitment to the game than might actually be necessary in an ambient game implemented in an ambient intelligence environment.

In addition to supplying the distance walked, there are two ways for players to control avatars. Firstly, the avatars move in random directions, without player intervention, automatically fighting enemies they encounter, and so on. Secondly, the player may decide the direction their player character travels, and hence determine the occurrence of fights, and so on. These directions are supplied to the researcher with the distance walked for entering into the game engine. For convenience, these two modes of play may be termed *passive* and *active*.

The virtual game world comprises a two-dimensional grid. Player characters can move north, south, east, or west (not diagonally). If they pass through a square containing a pickup, then this is automatically picked up and any actions triggered by the pick up are resolved immediately. If they pass through a square containing an enemy, then they automatically enter into combat, which is resolved by comparing player *attack* against enemy *defence*, modifying the outcome with a dice roll. If the enemy is defeated, then there is a chance it will drop something the player can take.

Ambient Quest was run at the Women in Games 2007 conference on the 19th and 21st April 2007 at University of Wales: Newport, United Kingdom. The conference was attended by academic game researchers and industry professionals. The delegates were all given pedometers, and computers were set up with the Ambient Quest program running

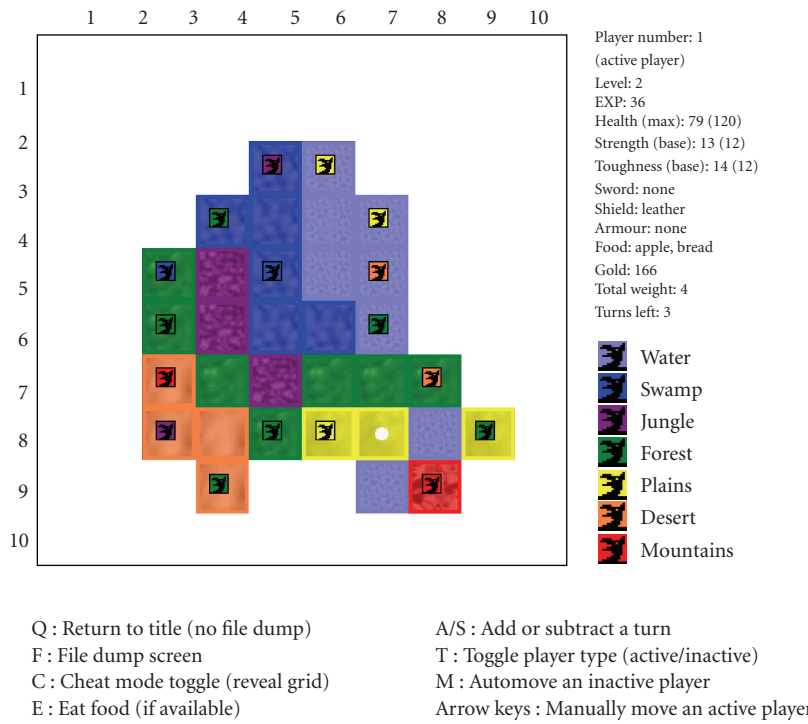


Figure 3: Ambient Quest screen layout.

on them so that distances moved could be converted into game moves.

The delegates quickly became engaged in the game, and it seemed to influence the mood at the conference with delegates talking about the game. The attention of players on the game varied greatly over time; they were able to ignore it or focus more fully on it at whim. Players also admitted to changing their behaviour, for example walking instead of getting a taxi. Players also *cheated* by shaking their pedometers. This was sometimes justified because pedometers had been accidentally reset. The pedometers had a reset button on the outside which was rather easy to press.

The positioning of the reset button resulted in *game modding* as players started pulling the buttons off the pedometers so they could not be accidentally reset. This pedometer modding was unexpected, emergent, game behaviour that had not been predicted. Over the second day of the game, players were talking about how far they had walked and discussing how far they *should* walk each day.

During the second day, a couple of people asked to have their moves put into the Ambient Quest game program, but this was a much smaller number than was expected.

Despite this low engagement with the virtual world, it was very clear from conversations with people that they were aware they were playing a game and that their movements would affect movements of an avatar. There was a very strong sense that people were engaged together in an ambient game as evidenced by their comments.

By the third day, it had become clear that the delegates were largely not interested in the process of transferring their pedometer readings into the Ambient Quest 2D world though they did continue to refer to the game. They already had a good idea of the gameplay, and it was as though they didn't actually need to go and see it played out on a screen. They could imagine it, which may not be surprising considering the conference delegates were mainly game researchers and developers.

One of the speakers at the conference, Julia Sussner, made an interesting observation about the way people were reduced to *steps*. The *steps* to *squares* transfer when using the pedometer reading to determine moves on a 2D map involved a numerical to graphical transformation. Perhaps there is a sense of enlargement when taking the scalar steps and using them to create vector moves. In his book *Little, Big*, Crowley [22] describes a world behind our world in which "the farther in you go, the bigger it gets". Beyond the moves on the 2D map there is an avatar with attributes, armour, weapons, supplies, gold, experience, and so on. The deeper the player moves into the game, the more complexity they will find, and the richer the game experience, which might lead to a greater influence on the real world .

Ambient Quest successfully managed to create a mood at the conference and to be as ignorable as it was interesting. The engagement of delegates lay on a spectrum from superficial to full engagement.

Would the reaction of the delegates have differed if they had just been handed pedometers without a game attached? Observations and conversations lead to the conclusion that the players would not have been as engaged; there was a definite buzz at the conference that something novel was being tried and that something was happening just out of sight, driven by the pedometers everyone was wearing.

Ideally, in a future version, Ambient Quest would be played in an ambient intelligent environment with a sophisticated 3D virtual world and intelligent interfaces, and many more aspects of the players' activities would be mapped onto their avatars, perhaps in a massively multiplayer version of the game. For example, a future implementation of Ambient Quest might use cameras embedded in the environment and face recognition software, and so forth to measure distance travelled.

9. THE FUTURE OF AMBIENT GAMES

Future research in ambient games is taking place to reveal not only their use as a form of entertainment but also their more serious uses, in line with the ideas mentioned previously on ambient utility games. There are plans to use biometric and locative readings to drive future ambient games.

If ambient games are proven to modify behaviour, then they might be designed to have a direct effect on productivity, rewarding productive work practices. They might also be used to enhance otherwise repetitive jobs, for example stacking shelves in a supermarket might lead to gains for virtual characters in an ambient game world.

Outside of the workplace, ambient games might be used to encourage healthy life style choices, perhaps increasing the amount of exercise that people take. The Ambient Quest game has been shown to affect simple choices such as whether to walk or take a taxi.

10. CONCLUSION

This article has described a novel way of playing games and has defined ambient gaming. The ambient music roots of ambient games have been described. The growth of technology suitable for implementing a full ambient game system has been described, and this has been contrasted and compared with existing game technologies including both augmented reality and pervasive gaming as well. A way forward for research into ambient games has been suggested with the description of playing a simple ambient game simulation, Ambient Quest, at a conference. Future applications have been suggested that might promote the idea that work can be play.

ACKNOWLEDGMENT

The authors specially thank Neil Dansey of Determined Software who programmed Ambient Quest.

REFERENCES

[1] M. Eyles and R. Eglin, "Ambient role playing games: towards a grammar of endlessness," in *Women in Games Conference*, Newport, UK, April 2007.

[2] M. Eyles and R. Eglin, "Entering an age of playfulness where persistent, pervasive ambient games create moods and modify behaviour," in *Proceedings of the 3rd International Conference on Games Research and Development (Cybergames '07)*, Manchester, UK, September 2007.

[3] B. Eno, *Ambient 1: Music for Airports*, EG Records, London, UK, 1978.

[4] B. Eno, "The Long Now-Transcript," November 2003, http://www.enoshop.co.uk/words.asp.

[5] E. Fredricksen, "Progress Quest (Version 6.2)," August 2004.

[6] R. Caillois, *Man, Play and Games*, University of Illinois Press, DeKalb, Ill, USA, 1961.

[7] E. Aarts, R. Harwig, and M. Schuurmans, "Ambient intelligence," in *The Invisible Future: The Seamless Integration of Technology into Everyday Life*, P. Denning, Ed., pp. 235–250, McGraw-Hill, New York, NY, USA, 2001.

[8] M. Weiser, "Ubiqutous computing," 1996 http://www.ubiq.com/hypertext/weiser.

[9] C. Weyrich, "Orientations for workprogramme 2000 and beyond," Tech. Rep., Information Society Technologies Advisory, Luxembourg, Belgium, September 1999.

[10] RFID Centre, "Introduction to RFID," 2005, http://www.rfidc.com/docs/introductiontorfid.htm.

[11] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: a literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, 2003.

[12] A. P. M. Grgic, "Face recognition," June 2006, http://www.face-rec.org.

[13] Human Pacman, "Human pacman," 2006, http://www.mixedrealitylab.org/research/HP/HP_webpage/research-HP-infor.htm.

[14] D. B. Thomas, "About the ARQuake Project," 2002, http://wearables.unisa.edu.au/Projects/ARQuake/www/index.html.

[15] D. A. Waern, "IPerG," 2006, http://www.pervasive-gaming.org/index_SWF.html.

[16] M. Chalmers, "Seamful design and ubicomp infrastructure," in *Proceedings of Ubicomp Workshop, at the Crossroads: The Interaction of HCI and Systems Issues in UbiComp*, Seattle, Wash, USA, October 2003.

[17] M. Bell, M. Chalmers, L. Barkhuus, et al., "Interweaving mobile games with everyday life," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pp. 417–426, Montréal, Québec, Canada, April 2006.

[18] S. M. Kangas, I. Outi, and C. Pöysä, "Ambient utility games: connecting utility to play," in *Proceedings of the International Conference on Internet and Multimedia Systems and Applications (EuroIMSA '05)*, pp. 18–24, Grindelwald, Switzerland, February 2005.

[19] J. Borland, "Blurring the line between games and life," February 2005, http://news.com.com/Blurring+the+line+between+games+and+life/ 2100-1024_3-5590956.html.

[20] "The alternating reality gaming network. What is an ARG?" September 2002–2006, http://www.argn.com/index.php.

[21] D. Sutch, "Fizzees (Physical Electronic Energisers)," 2006.

[22] J. Crowley, *Little, Big*, Methuen, London, UK, 1981.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 316790, 5 pages
doi:10.1155/2008/316790

Research Article

Efficient Terrain Triangulation and Modification Algorithms for Game Applications

Sundar Raman and Zheng Jianmin

School of Computer Engineering, Nanyang Technological University, Singapore 639798

Correspondence should be addressed to Sundar Raman, sund0010@ntu.edu.sg

Received 28 September 2007; Accepted 3 March 2008

Recommended by Kok Wai Wong

An efficient terrain generation algorithm is developed, based on constrained conforming Delaunay triangulation. The density of triangulation in different regions of a terrain is determined by its flatness, as seen from a height map, and a control map. Tracks and other objects found in a game world can be applied over the terrain using the "stenciling" and "stitching" algorithms. Using user controlled parameters, varying levels of detail can be preserved when applying these objects over the terrain as well. The algorithms have been incorporated into 3dsMax as plugins, and the experimental results demonstrate the usefulness and efficiency of the developed algorithms.

1. INTRODUCTION

The generation of a terrain is fundamental to many games today. Games like Tread marks [1] and Colin McRae Rally [2] were bestsellers mainly due to their impressive looking terrains, and by using technologies like dynamic LOD, continuous LOD, and deformable terrain over it. While all these are useful in solving specific problems related to rendering a huge terrain, they do not deal with the generation of a simple, yet realistic-looking terrain.

A terrain can be decomposed into two parts: (i) a triangular mesh, and (ii) one or more textures. In using such a terrain in a game, there are two main concerns: (i) the memory needed to store the terrain information for large worlds, and (ii) the time taken to load the big terrain into memory during a game's load-time. In this paper, the focus is on the triangular mesh part and algorithms are developed for its generation and modification.

Our terrain-generation algorithm is top-down, similar to Garland's approach [3], and it reduces the number of triangles required to represent the terrain for any particular level of detail, thus solving the two problems. The main difference between our algorithm and Garland's algorithm(s) is the introduction of a control map and several user parameters for refinement and minimization of triangle count. Our modification algorithm is based more on the real-world needs of a game artist, and uses the same triangulation technique to modify parts of the terrain. We have extended our previous work [4] to make it faster and more controllable.

The rest of the paper is organized as follows. Section 2 describes how to generate efficient triangular meshes for terrains using a height map, control map, and Delaunay triangulation. In Section 3, algorithms are proposed to easily modify the mesh so that objects found in a game, like tracks and guard rails can be applied over it. Two ways of doing this are discussed: (i) stenciling and (ii) stitching. The results for each section are shown separately. Finally the paper is concluded in Section 4.

2. TERRAIN GENERATION

The inputs of our algorithm are a height map, a control map and user defined parameters. A controlled, constrained conforming Delaunay triangulation algorithm is then used to create a triangular mesh that defines the terrain. Below is a description of each component of the procedure.

2.1. Height map

A *height map* is used to store elevation data for the terrain. The height map is a grayscale image where a white pixel

indicates the highest point, a black one the lowest point and shades of gray for heights in between. Each pixel is usually made up of 8 or 16 bits, thereby allowing us to store 2^8 or 2^{16} different-height values. Figure 2(a) shows a height map of a simple terrain and Figure 3(a) shows its corresponding triangular mesh representation.

2.2. Control map

Besides the height map, a *control map* is introduced, indicatingthe *regions in the terrain where more detail needs to be added*. The control map is also a grayscale image, where whiter regions are given more importance than the darker regions (refer to Figure 2(b)). Demarcating control regions has many uses, such as densely triangulating only those areas where the camera focuses often, like the area where we have tracks. Such an example is shown in Figure 6.

2.3. Delaunay triangulation

A *Delaunay triangulation* (DT) of a set of points P in a plane is a triangulation DT(P) in which no point $p \in P$ lies inside the circumcircle of any triangle in DT(P). The advantage of DT is that it produces a more *regular*-looking triangulation for any given point set, by maximizing the minimum angle of the output triangle set.

A planar straight line graph (PSLG), which is a collection of vertices and edges where endpoints of each edge are vertices of the PSLG, is used to initially represent the terrain. This is because the terrain needs to have well-marked boundary edges. Since the input is a PSLG, its triangulation becomes a *constrained Delaunay triangulation* (CDT). We also need to add more vertices on the inside and on each edge of the PSLG, making it a *constrained conforming Delaunay triangulation* (CCDT). The additional vertices inserted are called Steiner points [5].

The most common algorithms for DT are the *divide-and-conquer* [6], *sweepline* [7] and *incremental* [8] algorithms. Of these, Guibas and Stolfi's [9] implementation of divide-and-conquer algorithm has been found to be the fastest (see Su and Scot Drysdale [10]). Shewchuk [11] has adopted an optimization to this algorithm from Dwyer [12] to partition the vertices with horizontal and vertical cuts, thereby making it faster. Hence the same has been used in our implementation.

For constructing the CDT, first, the DT of the vertices is first performed; then, each missing input segment is forced into it by deleting the edges it crosses, inserting the segment and then retriangulating the two resulting polygons using the *ear-cutting algorithm* [13]. The CCDT is constructed using a variation of Chew's second algorithm [14], optimized for terrain generation using some assumptions (only boundary edges are constrained, etc.).

2.4. Terrain generation algorithm

The workflow of the proposed CCDT algorithm is shown in Figure 1. The algorithm generates the final mesh using a top-down approach. It begins with the four corner vertices of the height map, which determine the terrain's size. A rectangular PSLG is constructed, and an initial CDT simply splits the rectangle into two triangles.

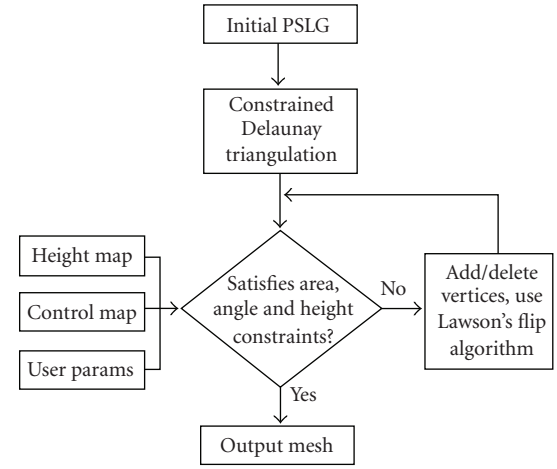
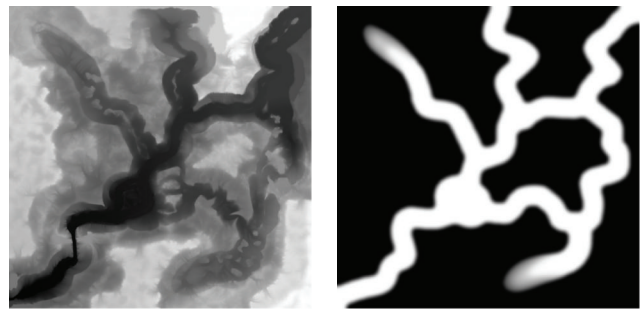


Figure 1: Proposed terrain-generation algorithm.

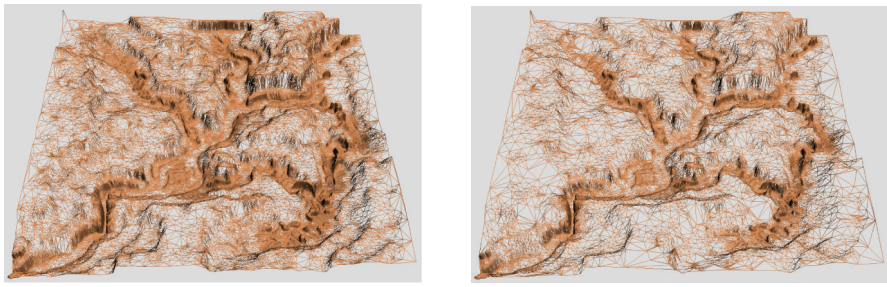


(a) (b)

Figure 2: Inputs to the terrain-generation algorithm (a) height map (b) control map.

Each triangle is checked to see if it satisfies the MinArea and MinAngle constraints. If it does not, it is rejected; else the more complex height constraint check is performed.

For the height constraint check, 1–40 equally distributed pixels are chosen inside each triangle, the exact number depending on the SkipScale parameter and the size of the current triangle. Next, all the pixels are checked to see if they satisfy the height constraint by comparing them with the height map and control map. If the control map is not specified, a default all-white or all-black control map can be assumed (we use all-black). If all the pixels satisfy the constraint, the triangle is accepted; otherwise, it is rejected. The rejected triangles are eliminated using the variation of Chew's second algorithm [14], by adding and/or deleting Steiner points, and maintaining the Delaunay condition using Lawson's *flip algorithm* [8]. The newly created triangles are checked as well, until all the triangles thus formed satisfy all three constraints.



(a) (b)

Figure 3: Output terrain mesh (a) hi-res with 175 K triangles, (b) low-res with just 75 K triangles.

2.5. User parameters

The user parameters shown in Figure 1 control the results of the output terrain. Eight parameters are introduced: SizeX, SizeY, Height, MinArea, MinError, MaxError, MinAngle, and SkipScale.

SizeX and SizeY specify the length and width of the terrain. Height is the maximum height of the terrain (all the height values from the height map will be mapped from 0 to this value), MinArea is the minimum area of any triangle in the projected 2D mesh. The height error in the control region is guaranteed to be less than or equal to MinError, whereas for the other regions, it is guaranteed to be not greater than MaxError. The algorithm further guarantees that the angle of any triangle found by projecting the mesh onto 2-dimension is greater than or equal to MinAngle. Ruppert [15] has shown that the upper limit for MinAngle is 20.7°, but Shewchuk [11] has argued that in practice, adding new vertices fails only when the angle exceeds 33.8°. To be safe, we have put an upper limit of 33° for MinAngle. SkipScale determines the overall accuracy of the algorithm.

These parameters, in addition to the control map, give a lot of flexibility to our triangulation algorithm, allowing users to generate different parts of the terrain at different resolutions.

2.6. Results

The algorithm was implemented as a 3dsMax plugin to generate the terrain. Figures 2(a) and 2(b) show the input height map and control map, respectively. Our algorithm generates a terrain with such a property (P) that the control regions and those parts where there are a lot of height variations are tessellated more densely, whereas fewer triangles are used for the relatively flat and noncontrol regions.

The terrain in Figure 3(a) was generated with the following parameters: SizeX = SizeY = 15840, Height = 2000, MinArea = 10, MinError = 5, MaxError = 50, MinAngle = 0, SkipScale = 1. Furthermore, by modifying the error parameters, a terrain of any resolution can be generated, with reduced number of triangles, while maintaining the property P at the same time.

For the terrain in Figure 3(b), MinError and MaxError were set to 10 and 100, respectively, with all the other parameters being the same. As we can see, this terrain looks reasonably detailed in the "nonflat" and control regions, but the other areas are greatly simplified to produce a mesh with 58% reduction in the number of triangles. This gives great flexibility to the artist who can generate multiple versions of the same terrain (using different control maps), which can be used in the game under different rendering contexts.

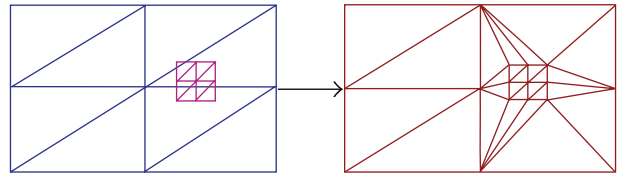


Figure 4: Stenciling a plane over a bigger plane.

3. TERRAIN MODIFICATION

After generating the Delaunay triangulated terrain, artists or programmers may need to modify it to overlay patches, tracks, and so forth, over it. This section describes two methods (stenciling and stitching) for such modification. Terminology for the sake of clarity: DM is the destination mesh SM is the source mesh. Our objective is to overlay SM over DM, at any specified position and orientation.

3.1. Stenciling

In *stenciling*, priority is given to the geometry of SM when overlaying it on DM. The amount of priority given is determined by an error parameter. To illustrate, a simple example as in Figure 4.

Let DM and SM be two simple 2×2 grids, with SM being smaller and positioned over DM as shown in Figure 4 (left). On the right, there is a single stenciled mesh in which

(i) all the edges and vertices of SM are projected and retained;
(ii) no new vertices are introduced;
(iii) the edges of DM which intersect with any edge of SM are removed;
(iv) the region around the stenciled part is retriangulated.

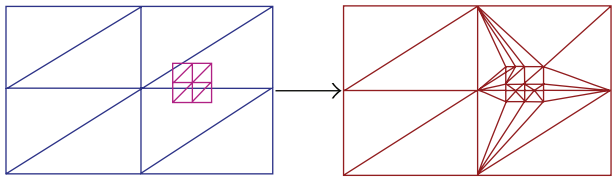


Figure 5: Stitching a plane over a bigger plane.

It is notable that SM is projected over DM "as-is." However, if DM is a terrain and SM is a track, this has disadvantages because we will lose height information pertaining to DM when vertices of DM inside the projected area are removed. Therefore, an error parameter is introduced, which controls how much height difference error is allowed over the stenciled region. This allows great flexibility to the artists who can make the track as detailed as they want to. A real life terrain-track example is shown in Section 3.3.

The main steps of the algorithm are as follows:

Stenciling algorithm

(1) Create a new empty mesh **M**
(2) Create a new PSLG **X**, initialize it to DM
(3) Generate height map **H** from SM
(4) Add projected SM vertices lying within DM to **X**
(5) Remove DM vertices lying within projected SM region.
(6) Remove all DM edges which intersect with projected SM edges, from **X**
(7) If SM extends beyond DM, calculate points of intersection of projected SM edges with bounding edges of DM, and add them to **X**
(8) Add all projected "full" SM edges (excluding those which lie fully or partially outside DM) to **X**
(9) Retriangulate
    (i) Apply the CCDT algorithm, set **M** = CCDT(**X**).
    (ii) Perform area, angle, and height constraint checks only for newly formed triangles lying within projected SM region
(10) For every new point (not belonging to DM) added to **M**, calculate its new height from **H** (from step 3)

For checking constraints, parameters can be set from the user interface, similar to terrain generation.

3.2. Stitching

Stitching is the "error-free" and more intuitive way of overlaying SM over DM. True to the word, the two meshes are simply "stitched" together, while following the height of DM. To illustrate, the same example as in Figure 4:

As compared to stenciling, the right of Figure 5 shows a mesh in which

(i) all the projected edges of SM are either retained or split;
(ii) all the edges of DM are also retained or split;
(iii) new vertices are created wherever DM and SM intersect;
(iv) the region around the stitched part is retriangulated.

The advantage of stitching is that no "new" vertices are introduced by the triangulation algorithm, except for the newly created intersection points. This implies that the height difference error is always zero, and we get a perfect stitched mesh. It also means there is no need of any user parameters for angle, area, and height error constraints, since no more triangles than necessary are created.

The stitching algorithm is outlined below:

Stitching algorithm

(1) Create a new empty mesh **M**
(2) Create a new PSLG **X**, initialize it to vertices of DM
(3) Add all nonduplicated vertices of SM lying within DM ($\mathbf{V_S}$), to **X**
(4) If SM extends beyond DM, calculate points of intersection of projected SM edges with bounding edges of DM ($\mathbf{V_B}$), and add them to **X**
(5) Add new vertices created due to intersection of every projected SM edge with every DM edge ($\mathbf{V_N}$), to **X**
(6) Add all unaffected edges of DM (not intersecting with any projected SM edge), to **X**
(7) Add all unaffected projected edges of SM (not intersecting with any DM edge) lying within DM, to **X**
(8) Add newly formed edges found by splitting DM edges to **X**
(9) Add newly formed edges found by splitting projected SM edges to **X**
(10) Retriangulate
    (i) Apply the CDT algorithm, set **M** = CDT(**X**)
(11) Calculate precise height of all new vertices ($\mathbf{V_S UV_B UV_N}$) by point-inside-triangle tests and interpolation

3.3. Results

Many experiments were conducted to test the algorithms, with terrains of different sizes as destination meshes and objects like tracks and patches representing plough lands as source meshes. Figure 6 shows a track object positioned over a terrain, at the exact orientation in which it needs to be applied. Figure 7 shows the effect of stenciling and stitching of the track over this large terrain.

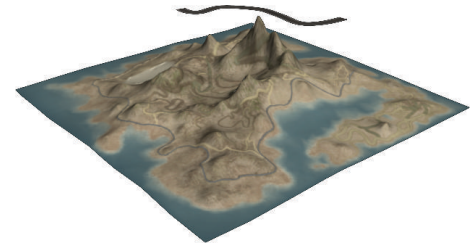


Figure 6: Input terrain and track meshes.

Figure 7: Track over terrain using stenciling.

Figure 8: Track over terrain using stitching.

It can be seen from Figure 7 that in stenciling, as few vertices are added "inside" the projected source mesh, the exact number controlled by an error parameter. For stitching, however, all the points of intersection are added, and hence there is no loss of height information in the final mesh (Figure 8). In other words, the track mesh perfectly follows the terrain and its height.

4. CONCLUSION

In this paper, an efficient terrain-generation method was introduced, based on the constrained conforming Delaunay triangulation, using a height map, control map, and some control parameters.

By using a height map together with a control map, generating a terrain using a top-down approach was automated to a large extent, providing game artists the means to create a terrain at different resolutions in different regions. Enhancements to the mesh were done via user parameters which specified angle, area, and height constraints. The nature of the chosen triangulation algorithm (Delaunay triangulation) further guaranteed the maximization of the minimum angle, hence producing a well rounded, more geometrically balanced terrain.

Two methods were proposed to modify the generated terrain: stenciling and stitching, to add objects such as tracks over the terrain. In stenciling, the geometry of the source mesh was retained as closely as possible, and the CCDT algorithm was used for retriangulation. In stitching, the source mesh followed the destination mesh perfectly, by calculating all points of intersection with it, and the CDT algorithm was used for retriangulation.

ACKNOWLEDGMENTS

The authors wish to express their sincere thanks to gameLAB Annexe, NTU, Singapore and TQ Global, Singapore, for providing them with excellent infrastructure and an ideal work environment for their research.

REFERENCES

[1] Tread Marks, Longbow Digital Arts Incorporated, 1999, http://www.ldagames.com/treadmarks/.

[2] Colin McRae Rally 04, The Codemasters Software Company Limited, 2004, http://www.codemasters.com/games/?gameid=1361.

[3] M. J. Garland and P. S. Heckbert, "Fast polygonal approximation of terrains and height fields," Tech. Rep. CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pa, USA, 1995.

[4] S. Raman and J. Zheng, "Efficient terrain triangulation and merging of world objects for game applications," in *Proceedings of the 3rd International Conference on Games Research and Development (CyberGames '07)*, pp. 46–51, Manchester, UK, September 2007.

[5] J. Steiner, "Questions proposées. Théorèmes sur l'hexagramum mysticum," *Annals of Mathematics*, vol. 18, pp. 339–340, 1827.

[6] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Parallel Programming*, vol. 9, no. 3, pp. 219–242, 1980.

[7] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, no. 1, pp. 153–174, 1987.

[8] C. L. Lawson, "Software for C^1 Surface Interpolation," in *Mathematical Software III*, pp. 161–194, Academic Press, New York, NY, USA, 1977.

[9] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 74–123, 1985.

[10] P. Su and R. L. Scot Drysdale, "A comparison of sequential Delaunay triangulation algorithms," in *Proceedings of the 11th Annual ACM Symposium on Computational Geometry (SCG '95)*, pp. 61–70, Vancouver, BC, Canada, June 1995.

[11] J. R. Shewchuk, "Triangle: engineering a 2D quality mesh generator and Delaunay triangulator," in *Proceedings of the 1st ACM Workshop on Applied Computational Geometry (WACG '96)*, pp. 124–133, Philadelphia, Pa, USA, May 1996.

[12] R. A. Dwyer, "A faster divide-and-conquer algorithm for constructing delaunay triangulations," *Algorithmica*, vol. 2, no. 1, pp. 137–151, 1987.

[13] H. Elgindy, H. Everett, and G. Toussaint, "Slicing an ear using prune-and-search," *Pattern Recognition*, vol. 14, no. 9, pp. 719–722, 1993.

[14] L. P. Chew, "Guaranteed-quality mesh generation for curved surfaces," in *Proceedings of the 9th Annual Symposium on Computational Geometry (SCG '93)*, pp. 274–280, San Diego, Calif, USA, May 1993.

[15] J. Ruppert, "A Delaunay refinement algorithm for quality 2-dimensional mesh generation," *Journal of Algorithms*, vol. 18, no. 3, pp. 548–585, 1995.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 753584, 7 pages
doi:10.1155/2008/753584

Review Article

Real-Time Optimally Adapting Meshes: Terrain Visualization in Games

Matthew White

Department of Computing and Mathematics, Manchester Metropolitan University, All Saints, Manchester M15 6BH, UK

Correspondence should be addressed to Matthew White, mattwhite06@googlemail.com

Received 27 September 2007; Accepted 21 December 2007

Recommended by Kok Wai Wong

One of the main challenges encountered by interactive graphics programmers involves presenting high-quality scenes while retaining real-time frame rates on the hardware. To achieve this, level-of-detail techniques can be employed to provide a form of control over scene quality versus performance. Several algorithms exist that allow such control, including the real-time optimally adapting mesh (ROAM) algorithm specifically aimed at terrain systems. Although ROAM provides an excellent approach to terrain visualization, it contains elements that can be difficult to implement within a game system. This paper hopes to discuss these factors and provide a more game-orientated implementation of the algorithm.

1. INTRODUCTION

Efficiently rendering meshes within a virtual environment requires the use of a level-of-detail (LOD) algorithm. This helps ensure that the number of primitives (triangles) used to represent the mesh is kept as close to an "optimal" level as possible. As graphics developers, we measure this level as a compromise between both scene detail (triangle count) and frame rate. The optimal level is then defined as the highest number of triangles we can render, while retaining an acceptable frame rate for our application.

Traditional level-of-detail methods begin by defining several versions of the scene's meshes, each differing in triangle count. As the application renders the scene, a version of each mesh is chosen in relation to factors, such as the meshes' onscreen size and overall scene importance. As meshes become closer or further from the viewer, their onscreen size changes and thus the number of triangles required to render them effectively. The result is a form of control over the scene triangle count and thus a more optimal detail level of the scene.

However, when applied to "massive" meshes, such as terrains, this technique breaks down. By massive, we mean a mesh whose size is so large that it is common for it to contain both very close and very distant sections from the viewer at one time. Simply put, we cannot just pick a distance from this vast range, apply a single detail level across the entire landscape, and expect reasonable results. Instead we need to implement a more specialized LOD algorithm that takes this range of distances into account. One of the first of these methods was introduced by Lindstrom in his continuous level-of-detail (CLOD) paper [1], which was then expanded upon by Duchaineau to produce the original ROAM algorithm [2].

ROAM works by defining a mesh as a hierarchal bintree structure of renderable triangles, dubbed by Duchaineau as a *binary triangle tree*. In this tree, each node represents a triangle that is a lower detail version of its two children nodes. Leaf nodes represent the highest LOD's triangles, while the root node represents the lowest. The rendering procedure then becomes a recursive task where we transverse the tree and decide which nodes to render for the current frame. When testing each node, we can choose to either tag the relevant triangle to be rendered this frame, or step a level deeper into the tree, and perform the same test upon the child nodes. Because each node represents 3 vertices (a triangle), a 3D location in the virtual world can be defined for the node and thus a distance from the viewer can be found. With this distance, we can perform the same distance test as the more traditional LOD algorithms, except that this test is now performed at a per-triangle level instead of the entire mesh. The result is that we can spread the LOD across the entire

visible terrain and thus solve the problem of the "massive mesh."

Although ROAM produces a range of detail levels for a terrain that can be tweaked to a specific triangle level, the algorithm itself does not translate to graphics hardware that well [3]. Because the graphics processor can only process data in its local graphics memory, any change to the renderable dataset requires an upload to this graphics memory. This upload can be considered expensive, and overuse of it can result in a problem known as "thrashing," causing the graphics processing unit (GPU) to stall as it waits for graphics memory to be written to. For high performance graphics, we prefer to load the required data onto the graphics card at initialisation time, and then attempt to minimise any further uploads during the runtime of the application. Duchaineau's ROAM relies heavily on changes to the mesh vertices, which are built to describe the current tessellation of the mesh. Uploading this buffer to the graphics memory can cause the mentioned thrashing effect and thus a performance hit, something that has caused criticism from games developers and led to simplified variations of the algorithm appearing in several games [4, 5].

Many of these variations have one thing in common. Instead of checking every triangle of the mesh for a correct LOD, the mesh is split into a collection, usually a grid, of terrain tiles. Each tile then contains several sets of geometry, each representing a different LOD for the tile, much like the more traditional LOD algorithms. Because each tile has a finite number of detail levels, they can all be uploaded to the graphics memory at initialisation time, minimising the thrashing effect. Therefore, better performance can be obtained by these ROAM variations, which can make them more desirable for games applications.

This performance increase has its cost however. By replacing the per-triangle LOD test with per-tile tests, we lose the tessellation accuracy of the algorithm. No longer can we increase or decrease the triangle count by a single triangle, and thus lose the near-perfect optimal detail level provided by the original ROAM method. Also, the effect known as "popping" can become much more apparent in these variations. Popping is the graphical artifact created when a visible part of the terrain changes its detail level. The geometry literally changes in front of the users eyes, and can become very distracting if large areas of the landscape suddenly switch. This effect is unavoidable, but can be reduced if the changing sections of the terrain are relatively small on screen. Since ROAM tessellates on a per-triangle basis, this area is usually sufficiently small for combating popping, but when entire terrain tiles change LOD, the effect can be much more noticeable.

Not all game systems that use ROAM implement this style of approach however. Treadmarks [6], an action game by Longbow Digital Arts, is probably the most well known of games that implement ROAM-based terrain. Instead of using the simpler versions of the algorithm, like those mentioned in Snook's and Ulrich's papers, Treadmarks uses a split-only approach, along with a technique called *Implicit Binary Trees* to increase performance [7]. Split-only means that the terrain's detail level is recalculated for each frame, without the *frame coherence* feature mentioned in Duchaineau's original paper. Although this requires more per-frame processing time, it greatly simplifies the algorithm making it much easier and quicker to implement into a game system.

The remainder of this paper will describe a new variation of ROAM that combines the ideas discussed in these previous variations into a new system, aimed mainly towards games and real-time graphical applications.

2. OVERVIEW

Originally presented previously at the Manchester CyberGames Conference [8], the "GEOmancy" terrain engine uses a version of the ROAM algorithm that overcomes the problems discussed. The system works by dividing the terrain geometry into a collection of tiles, each represented by a pair of ROAM triangle bintrees. The classic ROAM split-merge algorithm is then applied to each tile individually to produce an optimal detail level. To retain speed through hardware optimisations, the vertex buffer for each tile remains static and is uploaded to graphics memory at the application's start. The detail level of each tile is then described, instead, via an index buffer, which is created through transversing the tile's bintrees. Because the per-frame change in viewpoint position is usually a small fraction of the terrain size, the amount of LOD changes is also very small, resulting in very few updates to the separate tiles' index buffers. This allows the accuracy of the original ROAM algorithm to be maintained, while minimising the amount of data that must be posted to the graphics device per frame.

Although the algorithm tries to provide both high accuracy and high performance, it is liable to two major limitations. First, the algorithm only works on grid-based terrain geometry. That is, vertices that are spaced along the x-z plane at regular intervals with only their height values differing. This is not too much of an issue for games as this is by far the most popular terrain representation method, allowing the dataset to be compressed to a map of height values (a heightmap) and a single float that defines the distance between vertices. Secondary, due to the use of static vertices, only heightmaps of specific sizes can be used.This limitation can be overcome by using the next largest viable size and "voiding" off the unwanted extra vertices with water or walls, and so forth. The geometry may still be there, but techniques can be used to ensure that the player never sees it.

3. IMPLEMENTATION

3.1. Tiled geometry

The GEOmancy algorithm begins by converting a heightmap into a grid of terrain tiles. For each tile, a vertex array is created by sampling the relevant heightmap entries and scaling these values to produce terrain heights and thus vertices. These vertex arrays can then be placed in the graphics memory ready for future render calls.

For each tile, we need to create two bintrees, each represented by an index buffer. When we tessellate our bintree, this index buffer will contain a description of which triangles to render to provide the current LOD of the tile. As stated

previously, each node of a triangle bintree represents a renderable triangle. Because we are using an index array to reference which vertices to render, a triangle can be represented using three integers that can be used to index the appropriate vertex array. As well as this, we also need to store an error metric for the triangle, similar to Duchaineau's ROAM, so that we can perform LOD tests at runtime for each node in the tree. Since we cannot know the distance to the viewpoint at initialisation time, we need to store a value that can assist us during the runtime LOD decisions. For this, a technique from the Treadmarks engine is used called variance.

Since every non-highest detail level triangle is an approximation of its children, a difference for it can be calculated by finding the distances between it and the actual height of the geometry that it covers. When we run our LOD tests, we can say that triangles with a high variance are bad representations of the geometry they cover, and should receive a higher "split" priority than those with lower variances. When our algorithm is deciding where to add triangles to the frame, the variance measure helps ensure that rougher sections of the terrain will receive more detail than the flatter parts, which is exactly what we require.

As stated previously, our terrain tiles must be of a specific size. This is because an existing vertex at the correct point is required to split a triangle in two. Because of this, only specific sizes will allow us to split triangles down to the lowest level possible. As can be seen in Figure 1, there is a limited number of tile dimensions that allow this situation.

For each increase in usable detail levels for a tile, we are required to double the number of triangles along their edges. The size of the tile, in vertices, required for this can, therefore, be defined as $[(2^n) + 1]$, where n is the depth of the tile bintrees. For the demo, we used a dimension of 9×9 vertices per tile, as it provided a good balance between bintree depth and number of tiles.

3.2. Implicit bintrees

Now that we have divided our terrain into tiles, we need to create our version of the ROAM triangle bintrees. As mentioned previously, we will be using an updatable index buffer to describe which triangles to render from our vertex array. To help boost performance, a technique, again from the Treadmarks engine, called Implicit Bintrees, will be used. Because our trees will never add or remove nodes after the initialisation phase, we can represent our bintrees through a fixed-sized array, providing an abstract interface that accesses it like a bintree. The result is that all memory allocations are done at initialisation, improving the performance of the runtime part of the algorithm. An excellent explanation of this process was presented by Bryan Turner on the Gamasutra website [9].

The first index of our array stores the root node of the tree. Transversing the tree can be quickly achieved via bitshift operations as follows.

Left-Child Index: curIndex $\ll 1$.
Right-Child Index: (curIndex $\ll 1$) + 1.
Parent Node: curIndex $\gg 2$.

These macros enable a parent or child index to be found from any other array index, through the use of very fast operations, as well as removing the need for each node to store pointers to its neighbours.

Perhaps the biggest advantage of implicit bintrees (other than removing the need for dynamic memory allocation), is that any triangle in the tree can now be described using a single integer index. As will be covered later, this fact is particularly useful for implementing the ROAM split and merge queues, as well as solving the CLOD problem known as cracks.

In our algorithm, each tile contains two of these implicit bintrees, one for the "top-left triangle" and one for the "bottom-right one." For each bintree node, we store three indices that describe the triangle vertices, along with a variance value for the triangle. We define our root node as a triangle with vertices at the relevant corners of the tile. Every child can then be defined by dividing the parent triangle down its centre, creating the two half-sized child triangles. This process can be repeated recursively through the tree to create all potential triangles for each tile.

3.3. Error metrics

To complete our bintrees, we need to find the variance value for each node of the tree. This is a recursive process that starts at the leaf nodes and works up to the root node. Because each leaf node represents our highest LOD, their variance value is 0. For each node above these leaves, we sample the height value from the heightmap where their hypotenuse's midpoint aligns to. We also find the average of the two hypotenuse's vertex heights to find the approximate rendered height at this point of the triangle. Variance is then the maximum of either the difference of these values, or of the two children's variance values. This max operation helps prevent a situation where a low detail triangle midpoint happens to fall at the same point as, or near to, the original height data. Whereas the variance for this would be near zero, the actual triangle itself could still be a bad approximation for the other points of the terrain that it covers.

At run-time, we can perform an error test per node based upon the relative variance value. To make the algorithm view-dependant, we take factors concerning the virtual camera into account when making this test. As mentioned, these factors are usually in relation to the triangles' onscreen size, and thus the viewpoint distance. A typical test divides the variance value with this distance and checks the result against a threshold. This ensures that closer, rougher terrain is split with more scrutiny than distant, flatter parts. If this test fails, then we can "split" the triangle by stepping down one level of the tree and repeating the test on the two child nodes. Once we find a node that passes the test, we can add the three indices stored for the triangle to the terrain tile's main index array. When all the tests have been completed, the tile's index array will describe an optimal tessellation of the mesh for that frame, and can be used to reference the vertex array when rendering.

Although this works and allows per-triangle tessellations on a frame-by-frame basis, it is not entirely performance

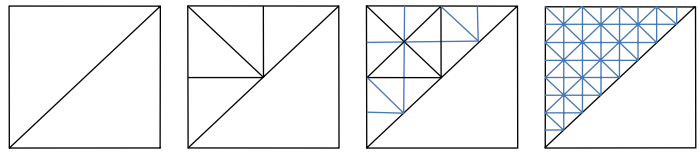


Figure 1: Static tile-size restrictions.

friendly. For each frame, we must recursively test every tile's bintrees from their root node to find the optimal detail level. Since our variance values do not change after initialisation, the only varying factor for our tests is the viewpoint itself. In most applications, it is not usual for the camera to move far between frames, so the results of the majority of tests will be identical to the previous frame's results. Therefore, instead of transversing the bintrees from root node down, we can "pick up" from where we left off last frame, testing each bintree from its previous optimal state. This optimisation is known as *frame coherence* and is a very effective part of the original ROAM algorithm.

3.4. Split-merge queues

In ROAM, frame coherence is achieved by using two queues called the split and merge queues. The split queue is used to store the next nodes that can be "split," thus increasing the bintree's effective LOD, whereas the merge queue stores nodes that can be "merged" to decrease the LOD. Splitting a triangle is the process of converting it into its two child triangles, and therefore incrementing the mesh's triangle count, whereas merging is the reverse process of converging two triangles into their parent.

Implementing these data structures is relatively straightforward. Because our system is using static vertex arrays, the number of potential triangles is also constant, and thus the maximum number of triangles that could be on either queue. We can therefore implement each queue as a fixed length array of this size, with each array containing the indices to relative nodes within the implicit bintree. We can then use markers to store the effective starts and ends of the active parts of these queues, and never have to reallocate memory during run-time.

These queues represent the detail level state for a single bintree. We create two operations that allow the increase and decrease of this detail level called Split and Merge, respectively. The following is the pseudocode for a typical implementation for these operations.

Split operation

Pop top node index from the Split Queue.
Push this index to the Merge Queue.
Find node's child indices using the implicit bintree bitshift macros.
Add child indices to the end of the Split Queue, in order of Variance values.

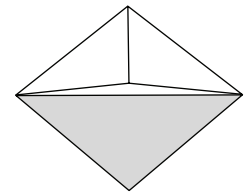


Figure 2: Cracks between triangles.

Merge operation

Pop top node index from the Merge Queue.
Push this index to the Split Queue.
Find node's child indices using macros.
Remove these child node indices from the Split Queue.

By restricting access to the queues to these 2 operations, we can ensure that the queues are always ordered by the node's variance values. This helps ensure that higher variance areas of a bintree are split and merged before the lower variance parts, which is exactly what we want.

Because the split queue of a bintree contains a list of all visible triangles, an index buffer for the tile mesh can be built up by iterating through it and referencing the appropriate structures directly. Furthermore, we can tag which bintrees have had their split or merge methods accessed for each frame, and only upload the new index buffer for them. Because the number of per-frame tessellations is usually a small percentage of the visible terrain tiles, this results in a great reduction in the amount of data being transferred to the graphics memory.

3.5. Avoiding cracks

3.5.1. Overview

One problem that all level-of-detail algorithms have to deal with is that of cracks appearing between different LODs. In ROAM, this occurs whenever a triangle is split. As can be seen from Figure 2, the extra vertex at the children's heads is at a different height than the second triangle, thus resulting in a gap. To solve this, the triangle at the base of the splitting triangle must also be forced to split.

There are three possible arrangements of triangles when splitting: base-to-nothing, base-to-base, and base-to-edge.

Base-to-nothing occurs when our split target triangle's hypotenuse (the base) is at the edge of the mesh, and thus no geometry. In this situation, we simply do nothing and split the triangle as normal.

Base-to-base is when the triangle shares its hypotenuse with its neighbour. In this situation, we simply force the neighbour triangle to split before the target triangle splits. The result is that the crack is covered up as the neighbour triangle's children reference the same vertex at that point.

Base-to-edge is perhaps the most complex scenario. In this case, our triangle's base-to-base partner is one of the neighbour's triangle's children. We essentially need to split twice, once for the neighbour and once for its appropriate child. The reason why this can become complex is that this initial split can encounter the same base-to-edge scenario as the original split. The result is that forced splitting can be propagated across the mesh, as triangles force other triangles to be split.

However, because a base-to-edge scenario can only occur between a triangle and a triangle of a detail level that is one less, this propagation seldom travels very far, so this rarely becomes an issue in practice.

To implement this forced split, each triangle needs knowledge of its *diamond partner*, which is the triangle in the mesh that shares a base-to-base relationship. With this knowledge, the triangle can inform its partner that it too needs to split. In the original ROAM, a pointer to this partner was stored on a per-triangle basis. However, in a terrain mesh that can contain hundreds of thousands of potential triangles, this extra memory requirement can soon mount up. GEOmancy takes advantage of the implicit bintrees and uses a *neighbour map* to significantly reduce these memory requirements.

3.5.2. Neighbour map

As mentioned previously, we can describe any potential triangle in our mesh with a pointer to a bintree and an index that defines which slot of the implicit bintree array to look at. We also know that, apart from the underlying height data, the structure of every bintree in our system is the same. The consequence of this is that every node in a bintree is also surrounded by, *relatively,* the same neighbours as simular nodes in other bintrees. With this similarity in mind, instead of storing a diamond partner pointer for each triangle, we can create a static *map* that when queried can return a description of the required partner.

Because a diamond partner shares its base with its partner triangle, it can only be in either the same bintree or a neighbouring bintree. During initialisation, we store three pointers for each bintree, each of which point to the relative neighbouring bintree. These pointers can even be null in the case that the tree is at the edge of the mesh. Upon querying, the neighbour map returns a partner's array index and also a flag that denotes which bintree neighbour the index refers to (left-edge, right-edge, or hypotenuse-edge neighbour). With this information, a bintree can call the split function through the relevant pointer, passing in the index to produce a forced split. In the case of a "same bintree" flag, the bintree class simply calls its own split method. In the event that a neighbour pointer is null, then the split can be assumed to be a base-to-nothing scenario, and the forced split can be ignored.

The end result is a fast look-up system for finding diamond partners that does not require per-triangle pointer storage. The neighbour map's size remains fixed regardless of the size of the terrain, which can prove very beneficial for systems that require vast landscapes.

Creation of the neighbour map is a recursive task much like the creation of the bintrees. It was found that, with the exception of the root triangle, every triangle's neighbours could be found by examining their parent. Root triangles have no parent node, so their neighbours must be defined upon the bintrees creation, through the use of the neighbour pointer class members mentioned previously. The neighbour map itself is an array of the same size as the system's bintrees. At each slot, we store a flag, indicating the root triangle's bintree neighbour, and another index, describing the specific triangle from this bintree.

Figure 3 shows the graphical representation of the first 3 levels of a bintree. As we can see, the bintree (triangle 0) has the neighbours L, R, and H denoting left-edge, right-edge, and hypotenuse neighbours, respectively.

The left child of this triangle is triangle 1. As can be seen, its neighbours are as follows:

(i) left neighbour: triangle 0's right child;
(ii) right neighbour: triangle 0's base neighbour;
(iii) base neighbour: triangle 0's left neighbour.

The right child of the root (triangle 2) shares a similar relationship:

(i) left neighbour: triangle 0's base neighbour;
(ii) right neighbour: triangle 0's right child;
(iii) base neighbour: triangle 0's right neighbour.

The next level down (triangles 3, 4, 5, and 6) follows the same pattern depending on if they are the left or right children of their parent. Using this information, we can use a recursive method to fill the neighbour array with a flag and index number, describing the relative location of the diamond partner for any node in one of our bintrees.

3.6. Summary

At the end of the initialisation, we have converted our heightmap into a grid of terrain tiles. Each tile represents a square of geometry of our terrain, represented via vertex buffers, and also two bintrees. These bintrees represent the current tessellation of the terrain tile, using split-merge ROAM to produce an index buffer that denotes which triangles to render from our geometry. These bintrees offer split and merge methods to increase or decrease the tree's LOD by a single triangle.

During run-time, we test each tile against an error threshold using both its split-queue's top node's variance and the distance to the tile from the camera. These factors insure that the worst approximations and the closest triangles are split at a higher priority.

To maintain an optimal level of performance, we also use frame-by-frame coherence offered by the split and merge queues. Because of this, and the segmentation of the geometry due to the tiled terrain, only a small proportion of the

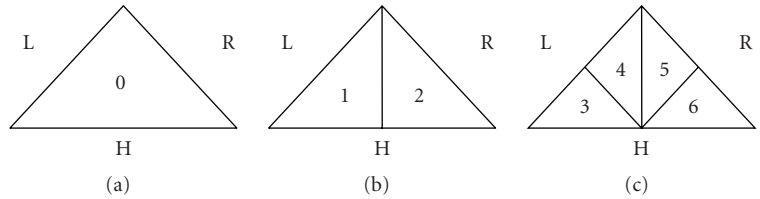


Figure 3: Neighbourhood map structure.

terrain's entire index buffer requires changing each frame, minimising the effect of thrashing.

Finally, any bintree can query the neighbour map for a description of a specific triangle's diamond partner, from which it can force another local bintree to split a triangle, avoiding cracks from appearing within the mesh.

4. RESULTS

To test the final implementation of the system, a sample heightmap was used and frame rates were observed. A heightmap of size 512×512 was chosen for these tests, and thus provided just over 522 000 potentially renderable triangles in the mesh. The system used for the tests was a typical desktop system; 2.0 GHz CPU, 512 MB RAM with an ATI Radeon 9600 graphics card. Different error metrics were tested to see the difference between performance and scene quality during the rendering. Table 1 shows the average frame rates achieved for several error metrics.

Table 1: Frame rates for specific error metrics.

Error metric	Average frame rate (per second)	Triangles (per second)
1.0	90.5	47.26 million
3.0	106.7	55.72 million
5.0	112.2	58.59 million
7.0	113.9	59.48 million

Metrics above 7 pixels provided slightly higher frame rates, but also suffered from very noticeable popping. By using a small error metric, this popping effect was restricted to the smaller onscreen triangles, and was not as noticeable. Without some extra feature to deal with these artifacts, however, metrics over 7 are unlikely to be favourable for use within a games application.

For comparison, the most recent version of ROAM (ROAM 2.0) shows a performance between 40 million and 56 million triangles per second [10] depending upon the hardware being used. Depending upon the error metric chosen, our system can produce higher frame rates while maintaining an acceptable level-of-detail. The full source and an executable demo for the GEOmancy system can be found online at http://members.gamedev.net/rootevilgames/mwhite/GEOmancy.htm.

5. FURTHER WORK

At the time of writing, the GEOmancy algorithm provides a new variation of ROAM, aimed for implementation within a games-orientated system. However, there are further improvements being worked on that will be discussed in this section.

Memory can be a tight resource, especially in the development of console games. Storing an entire dataset for a landscape can hog up much of this resource. To get round this, we intend to make as much of the vertex data reusable as possible. The idea revolves around the use of vertex buffer streams. In one stream, we load the vertices' x and z positions. Because these are repeated for each tile, due to the grid nature of the mesh, we can create a single vertex buffer for each tile to reference. A second stream can then be used to reference other vertex data, such as the y position and texture coordinates.

For systems using pixel shader 3.0, an expansion of this technique can be applied using vertex textures. This way, each tile's vertex height can be referenced directly from the heightmap texture, moving part of the processing onto the GPU. Normal maps can also be used in the same fashion to provide fast per-vertex normals for dynamic lighting.

Perhaps one of the most exciting ideas for future development is the incorporation of DirectX 10's new geometry shader. This is a shader stage that allows the generation of new primitives within the rendering pipeline itself. As mentioned in my previous paper, the main reason that there has been no GPU-only implementation of ROAM is the inability to add and remove vertices in this pipe-line. With this new shader, this limitation should no longer apply and the creation of a full GPU ROAM algorithm could soon become a reality.

6. CONCLUSION

ROAM is a popular and very effective algorithm for the visualisation of terrains. However, several problems and performance issues can be encountered when trying to implement it into a performance-heavy application, such as a computer game. This paper has presented an overview of the original algorithm and discussed a possible implementation of a more games-orientated variation. By imposing size restrictions upon the input geometry, memory requirements can be precalculated during the initialisation stages, eliminating the need for dynamic memory allocations at run-time. Finally, a tile-based system has been incorporated, allowing us to treat each terrain tile as a separate mesh. This allows us to separate the terrain mesh's index buffer into more manageable

sections, rebuilding only the parts that require it between frames.

ACKNOWLEDGMENTS

The author would like to thank Geoff Brindle for his assistance throughout his dissertation, which led to the development of this paper. He would also like to thank Edmund Prakash for general assistance during his dissertation, as well as allowing him to present his original paper at the Manchester CyberGames Conference 2007.

REFERENCES

[1] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner, "Real-time, continuous level of detail rendering of height fields," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 109–118, New Orleans, La, USA, August 1996.

[2] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, and M. Mineev-Weinstein, "ROAMing terrain: real-time optimally adapting meshes," Tech. Rep. UCRL-JC-127870, Lawrence Livermore National Laboratory, Livermore, Calif, USA, July 1997.

[3] G. Snook, *Real-Time 3D Terrain Engines Using C++ and DirectX 9*, Charles River Media, Hingham, Mass, USA, 2003.

[4] G. Snook, "Simplified terrain using interlocking tiles," in *Games Programming Gems 2*, pp. 377–383, Charles River Media, Hingham, Mass, USA, 2001.

[5] T. Ulrich, "Chunked LOD," http://www.tulrich.com/geekstuff/chunklod.html.

[6] Longbow digital arts, Treadmarks, http://www.ldagames.com/treadmarks.

[7] S. McNally, "Treadmarks Engine (Binary Trees and Terrain Tessellation)," http://www.ldagames.com/.

[8] M. White, "Adapting ROAM for use within a games application," in *Proceedings of the 3rd International Conference on Games Research and Development (CyberGames '07)*, pp. 59–66, Manchester, UK, September 2007.

[9] B. Turner, "Real-Time Dynamic Level of Detail Terrain Rendering with ROAM," http://www.gamasutra.com/features/20000403/turner_01.htm.

[10] M. Duchaineau, ROAM Algorithm Version 2.0, http://www.cognigraph.com/ROAM_homepage/ROAM2.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 135398, 7 pages
doi:10.1155/2008/135398

Research Article

Auto Coloring with Enhanced Character Registration

Jie Qiu,[1] Hock Soon Seah,[1] Feng Tian,[1] Quan Chen,[1] Zhongke Wu,[2] and Konstantin Melikhov[1]

[1] *Interaction and Entertainment Research Center, School of Computer Engineering, Nanyang Technological University, 50 Nanyang Drive, Singapore 637553*
[2] *College of Information Science and Technology, Beijing Normal University, Beijing 100875, China*

Correspondence should be addressed to Jie Qiu, jqiu@ntu.edu.sg

Received 27 July 2007; Accepted 2 October 2007

Recommended by Kevin Kok Wai Wong

An enhanced character registration method is proposed in this paper to assist the auto coloring for 2D animation characters. After skeletons are extracted, the skeleton of the character in a target frame is relocated based on a stable branch in a reference frame. Subsequently the characters among a sequence are automatically matched and registered. Occlusion are then detected and located in certain components segmented from the character. Two different approaches are applied to color regions in components without and with occlusion respectively. The approach has been tested for coloring a practical animation sequence and achieved high coloring accuracy, showing its applicability in commercial animation production.

1. INTRODUCTION

Inking/coloring of the individual animated characters for every frame is one of the most time-consuming and labor-intensive procedures in cel animation production. Many systems and some algorithms have been proposed to assist the coloring of cel animation production. However, these systems or algorithms have significant limitation, in terms of not being able to automatically establish the correct region correspondences when there is a big change or occlusion of the character, as reviewed in [1]. In summary, *computer-assisted auto coloring (CAAC)* remains a tough issue in research.

To detect and handle occlusion, a novel character registration method was proposed in our previous work [2], as the first attempt for auto coloring 2D characters for some special cases. As indicated in [2], the auto coloring approach may be challenged if the character's position changes greatly in a sequence of frames. In this paper, a relocation method is proposed to enhance the character registration approach in [2], making the auto coloring approach more robust.

The rest of the paper is organized as follows. First, the enhanced character registration method is introduced in detail. Subsequently, our occlusion detection and auto coloring process is introduced. Experiments are then designed to test the algorithm and results analyzed. Conclusion and future work can be found at the end of this paper.

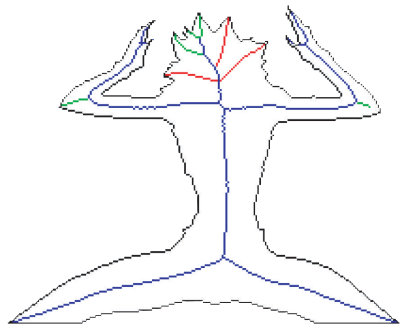
2. CHARACTER REGISTRATION

2.1. Skeleton extraction

A skeleton is a useful shape abstraction that captures the essential topology of an object in both two and three dimensions. It represents a thinned version of the original object, which still retains the shape properties of the original object [3]. The skeleton of a 2D character can preserve the most essential geometric and topological information of it, which makes it suitable for our character registration purpose.

In practical animation production, some characters are complex featuring sharp convex or concave details in their outline, which introduce redundant branches when the skeleton extraction approach based on active contours [4, 5] is applied. As shown in Figure 1, the branches in red and green do not relate to the physical structure of the character. These are viewed as "noise" in this paper. The short redundant branches in Figure 1 can be easily pruned based on a threshold, but it is difficult to remove the long ones. To reduce the noise, Gaussian filtering is applied to smooth the outline and remove the details. The Gaussian function in 2D form is

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{(x^2+y^2)/-2\sigma^2}. \quad (1)$$



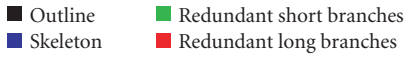


Figure 1: Extracted skeletons.

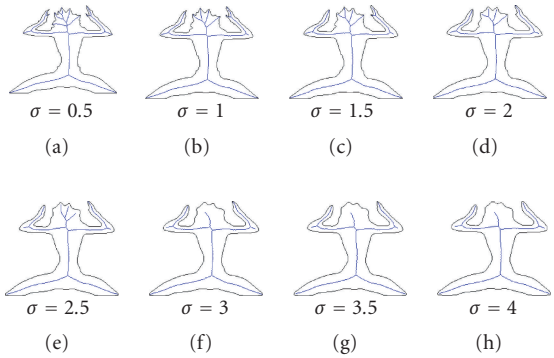


Figure 2: Skeleton extraction after Gaussian smoothing.

The smoothed outlines and corresponding skeletons with different σ are illustrated in Figure 2. The σ is empirically selected as 3 for the examples used in this paper.

After the outline is smoothed, the skeleton of each character is extracted using active contours approach [4, 5], and further thinned under SUSAN thinning rules [6], and pruned. Finally, it is segmented into *L-branches* and *J-branches* as introduced in [2]. Topology graphs of each character is also extracted using the approach introduced in [5]. Figure 3 shows the extracted skeleton and topology graph for the character in Frame 4 of the first test used in the paper. Junction nodes are illustrated in red, leaf nodes in blue, L-branches in green, and J-branches in black, respectively.

2.2. Skeleton relocation

Animation is an art of capturing a series of individual movements, whether on film or in digital form, and replaying them in rapid succession to give the illusion of movement [7]. Accordingly, even in two successive frames, the positions of the two characters may differ much. Besides affine transforms which are often used for depicting the character's global motion, deformation of the character parts usually exists for expressing the local motion like the movements of articulated parts. With these uncertainties, it is hard to find an accurate transform to locate the two characters in the same position with the most similarity. When drawing inbetweens, animators locate characters of two key frames by overlapping the most similar parts of the character in the same position. Mimicking this method, two skeletons can be relocated according to a *stable branch* which does not change much among the whole animation sequence.

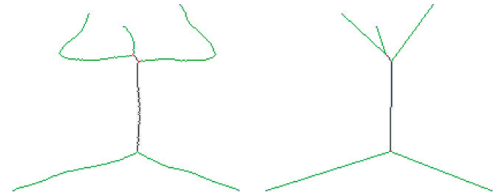


Figure 3: Extracted skeleton and topology graph.

The stable branch of each frame can be interactively selected by the user. Alternatively, user only needs to indicate the stable branch in the first reference frame to be registered. Given a reference frame and a target frame, a prediction and relocation method is proposed as follows to reduce user intervention.

Given a population of random vectors x, the principal components analysis (PCA) is defined as

$$y = A(x - m_x), \tag{2}$$

where A is the matrix whose rows are formed from the eigenvectors of x's covariance matrix C_x, m_x is x's mean vector [8].

(i) Compute the PCA transforming matrices T_i and M_i for the stable branch B_i in the reference frame, where T_i and M_i, respectively, correspond to A and m_x in (2).

(ii) Transform the skeleton S in the reference frame to the transforming coordinate system R_i^2 according to T_i and M_i:

$$S_i^t = T_i * (S - M_i). \tag{3}$$

(iii) For each branch $B_{i'}$ in the target frame, compute the dissimilarity value $D_t(i, i')$.

(a) Compute the PCA transforming matrices $T_{i'}$ and $M_{i'}$ for a branch $B_{i'}$ in the target frame, where $T_{i'}$ and $M_{i'}$, respectively correspond to A and m_x in (2).

(b) Transform the skeleton S' in the target frame to the transforming coordinate system R_i^2 according to $T_{i'}$ and $M_{i'}$:

$$S_{i'}^{\prime t} = T_{i'} * (S' - M_{i'}). \tag{4}$$

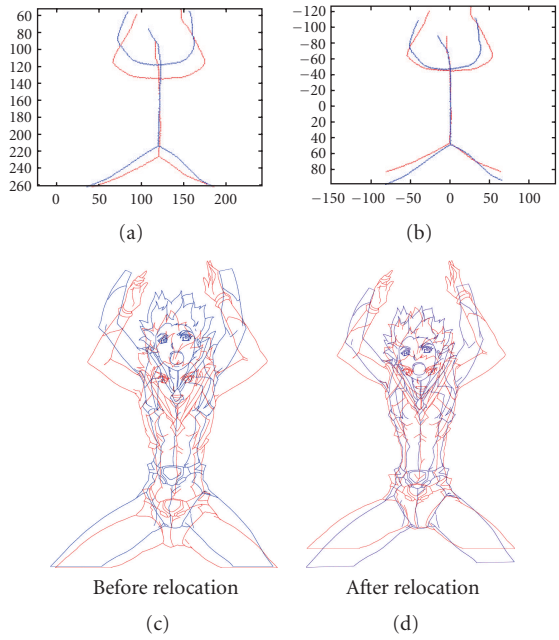


Figure 4: Skeleton relocation.

(c) Compute the Hausdorff distance $H_t(S_i^t, S_{i'}'^t)$ between S_i^t and $S_{i'}'^t$:

$$d(a,\ b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}, \quad (x, y) \in R_t^2,$$
$$\vec{h}(S_i^t, S_{i'}'^t) = \sup_{a \in S_i^t} \inf_{b \in S_{i'}'^t} d(a, b),$$
$$H_t(S_i^t, S_{i'}'^t) = \max\{\vec{h}(S_i^t, S_{i'}'^t), \vec{h}(S_{i'}'^t, S_i^t)\}. \tag{5}$$

(d) Compute the Hausdorff distance $H_t(S_i^t, -S_{i'}'^t)$ between S_i^t and $-S_{i'}'^t$.

(e) The dissimilarity value $D_t(i, i')$ between branches B_i and $B_{i'}$ is the minimum of $H_t(S_i^t, S_{i'}'^t)$ and $H_t(S_i^t, -S_{i'}'^t)$:

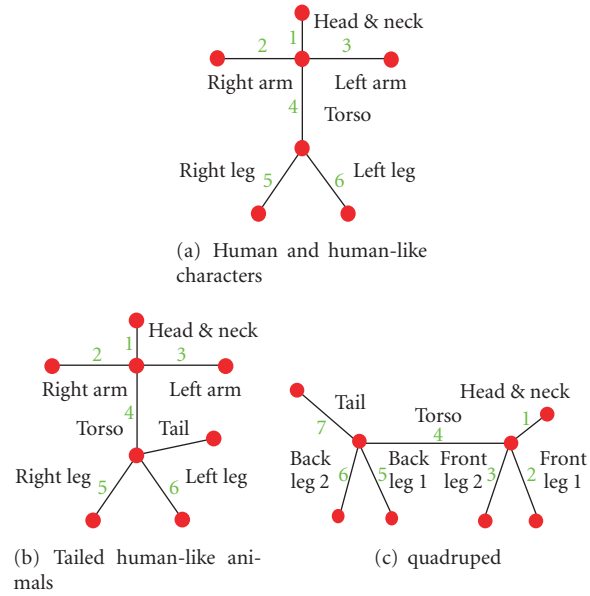
$$D_t(i, i') = \min(H_t(S_i^t, S_{i'}'^t), H_t(S_i^t, -S_{i'}'^t)). \tag{6}$$

(iv) If $D_t(i, i')$ is the minimum, B_i and $B_{i'}$ are corresponded, the skeleton S' in the target frame is transformed and normalized to be $S_{i'}'^t$ or $-S_{i'}'^t$ according to $T_{i'}$ and $M_{i'}$. The transforming coordinate system R_t^2 for S_i^t and $S_{i'}'^t$ or $-S_{i'}'^t$ is treated as the global coordinate system R_g^2.

Figure 4 shows the relocated skeletons and frames. Frames 1 and 2 are represented in blue and red, respectively.

2.3. Skeleton registration

General topology models are predefined based on the character's physical structure. Figure 5 illustrates some common topology models, which represent the most essential topological information of the characters.



(a) Human and human-like characters

(b) Tailed human-like animals

(c) quadruped

Figure 5: General topology model.

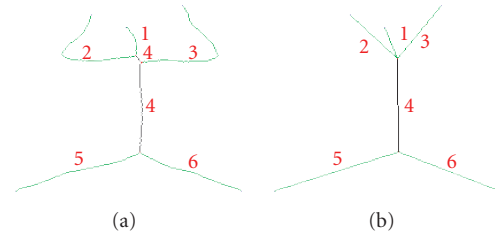


Figure 6: Skeleton registration.

To assure registration accuracy, a frame is selected as the first reference frame from the sequence to be painted, which contains a stable branch as introduced in Section 2.2 and the maximum number of branches among the sequence. The skeleton and topology graph of the character in the selected frame is then registered to the general model, as illustrated in Figure 6. The topology graph is adjusted accordingly.

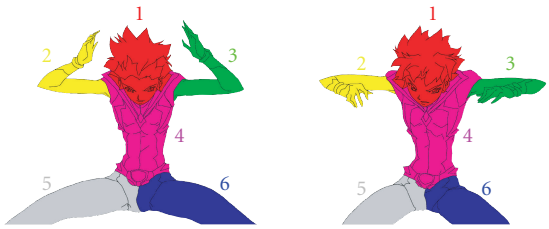
For characters in the other frames in the sequence, their skeletons and branches are registered based on the branch correspondences established by skeleton matching introduced in the next section.

2.4. Skeleton matching

As introduced in [2], our skeleton matching algorithm is based on both geometric and topological information of the skeleton, containing the following two steps.

2.4.1. Geometric matching

Global dissimilarity due to the motion of branches is represented by the global Hausdorff distance $H_g(B_i, B_{i'})$ and computed using the global coordinate system R_g^2 of the two frames.



Frame 4 (a) Frame 5 (b)

FIGURE 7: Segmentation result.

The local deformation of each branch is represented by the local Hausdorff distance $H_l(B_i, B_{i'})$ and computed using the local coordinate system R_l^2 of the two frames established based on PCA [2].

With the global and local Hausdorff distances between two branches, the dissimilarity value DV is defined as

$$\mathrm{DV}(i,i') = H_g(B_i, B_{i'}) + \min\left(H_l(B_i, B_{i'}), H_l(B_i, -B_{i'})\right). \tag{7}$$

Each branch B_i in the reference frame is matched with all branches in the target frame first. If $\mathrm{DV}(i,i')$ is the minimum among all dissimilarity values, a matching $(B_i \rightarrow B_{i'})$ is obtained. Subsequently, each branch $B_{i'}$ obtains its best-matched branch in the reference frame. If a bidirectional matching is achieved, $(B_i \rightarrow B_{i'}) \cap (B_{i'} \rightarrow B_i)$, B_i and $B_{i'}$ are corresponded $(B_i \leftrightarrow B_{i'})$, and $B_{i'}$ is registered as the same branch with B_i in the general topology model. The unregistered branches will be readjusted in later process.

2.4.2. Topological readjustment

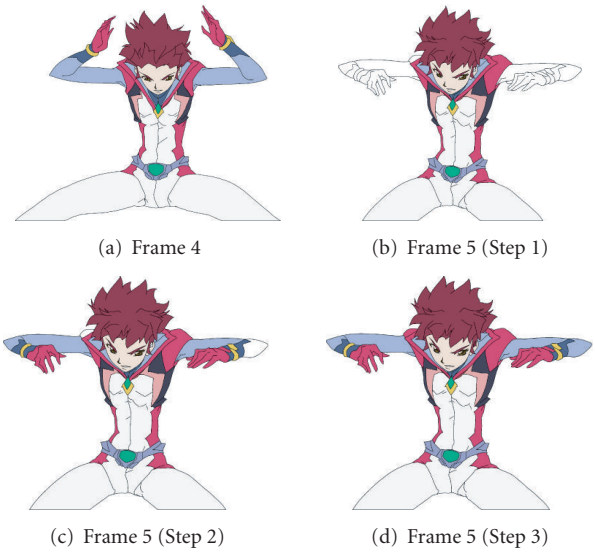
After geometric matching, some branches with big motion or deformation may be left unregistered. To match and register these branches and detect occlusion, a topological readjustment method is applied.

For each selected branch $B_{i'}^t$ in the target frame, its merging candidates are obtained, and it is merged with the one related most closely [2]. If no merging candidate is found, geometric matching is applied to those unregistered branches in $B_{i'}^t$'s subgraph.

2.5. Component segmentation

After skeleton registration, characters are segmented into several components corresponding to the branches in skeleton and topology graph, based on the explained area of each branch [2].

The segmented and registered components for Frames 4 and 5 are illustrated in Figure 7, where corresponding components are in the same color.



(a) Frame 4 (b) Frame 5 (Step 1)

(c) Frame 5 (Step 2) (d) Frame 5 (Step 3)

FIGURE 8: Auto coloring process.

3. ENHANCED AUTO COLORING

3.1. Occlusion detection

To detect and locate occlusion in components, a method based on the variation of region areas is advanced in [2].

(i) Each region in the reference frame is quantized and coded as an English character based on its area.

(ii) All the regions in the target frame are quantized and coded based on the scale points computed in the reference frame.

(iii) Regions in a component C_i in the reference frame are coded as a string S_i and compared with the string $S_{i'}$ of its corresponding component $C_{i'}$ in the target frame, and the least conversion cost between them is computed as $\gamma(S_i \rightarrow S_{i'})$.

(iv) Occlusion is located in $C_{i'}$ if $\gamma(S_i \rightarrow S_{i'})$ is bigger than an empirically defined threshold.

3.2. Auto coloring

With occlusion detection, regions can be divided into two categories: the first category contains all regions which are in the components without occlusion, and the second consists of those in the components with occlusion. The former is supposed to be more stable than the latter and different matching methods are applied to the two categories.

The coloring process consists of three steps, and the coloring result after each step is illustrated in Figure 8.

(i) For the first category, the hierarchical feature-based region matching approach as proposed in [1, 9] is applied.

(ii) For the second category, the continuity of some region contours is broken because of occlusion. Hence, features such as area, curve length, character points, and relations with neighboring regions change greatly. But

the rest of regions still have relatively stable features of area, curve length and character points. So a matching method similar to that for the first category is applied to match these regions. Each region is matched with those in corresponding components in the reference frame. The only difference is that the feature of relations with neighboring regions is ignored, so that the coded character string [1] is only composed of character points.

(iii) Finally, each of the remaining regions in the relocated target frame inherits the color of the region that it overlaps mostly in the reference frame. If no such region is obtained, which means that the region fully overlaps the background, it is painted in the color which fills the majority of the component (major color) that it belongs to.

After skeleton registration for the first reference frame, the prior and subsequent uncolored frames are selected as new target frames iteratively. The nearest colored frame which contains a stable branch is selected as the target frame's reference frame.

4. RESULTS AND ANALYSIS

To test the possibility of applying our approach in practical animation production, we use a Japanese-style sequence in practical production of the trailer of "Justeen" animation. The 8 frames as shown in Figure 9 compose 2/3 of a cut in the trailer, showing a boy jumping down into the control room of his robot. It can be noticed that occlusion arises due to the motion of the character's arms. To minimize the information loss due to occlusion, we select Frame 4 as the first reference frame. Then the prior and subsequent frames are colored one by one according to the color information of its next and prior frame, respectively. From the results we can see that most of the regions are correctly colored. The regions wrongly colored are mainly because no corresponding regions in the reference frame can be obtained due to the occlusion or information loss. Figure 10 illustrates the main errors among the sequence and the correct results. As illustrated in Figure 10(a), the right arm in Frame 5 changes greatly compared with the one in Frame 4, and the occluded part of the bracelet in Frame 4 is visible in Frame 5. Accordingly, the regions indicated in blue circles are wrongly matched as no correct correspondences exist in Frame 4. With more reference frames (which contain the corresponding regions) provided, this kind of error can be avoided. In Figure 10(c), the regions in blue circles in Frame 5 are created due to the occlusion, and they inherit the color of the regions they overlap mostly in Frame 4 as no corresponding regions are obtained. In Figure 10(e), the region in blue circle in Frame 7 is created as the hair is swaying to the position overlapping with the eyebrow. It is thus wrongly matched to a hair branch as they have great similarity. These errors can be solved if the hair is drawn in a separate layer. The mouth in Frame 1 is widely open so that the tongue appears only in this frame. It is not matched to any region, and thus inherits the color of the face in Frame 2. This kind of error due to information loss is hard to be avoided. In summary, the coloring accuracy for the total 7 uncolored frames is over 93%, which shows that our approach is applicable to practical animation production.

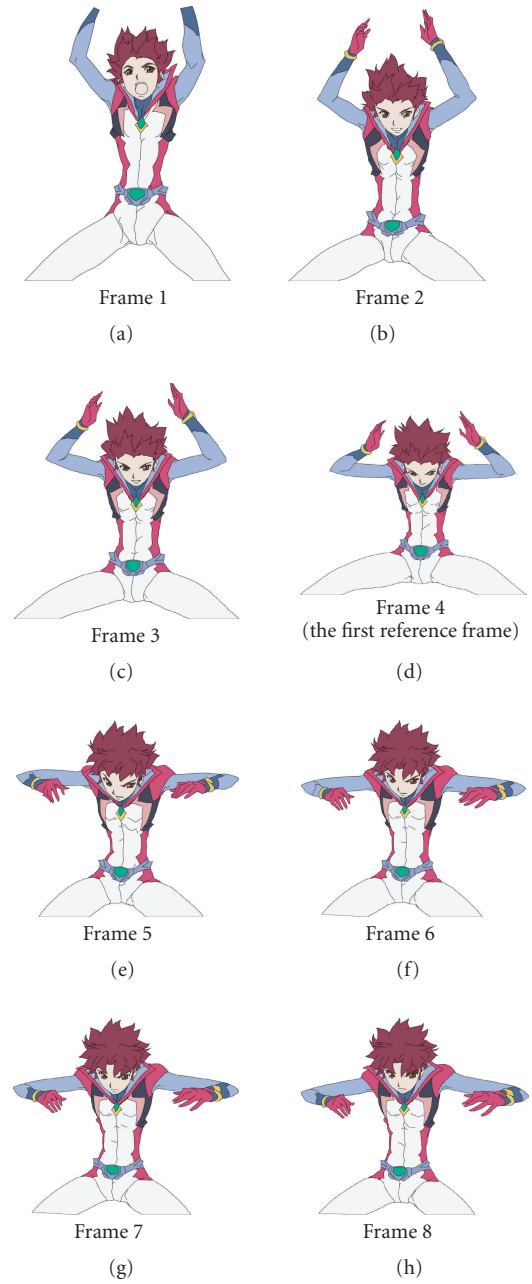


Figure 9: Test 1 (resolution: 2420 × 2420 pixels) (courtesy of Anime International Co., Inc., Japan).

Figure 11 shows the other test, with the first frame selected as the first reference frames. The character is simple, but it is noticeable that large motions and deformations exist among the sequence. With the relocation method proposed in this paper, skeletons are accurately matched, ensuring the high coloring accuracy for the test.

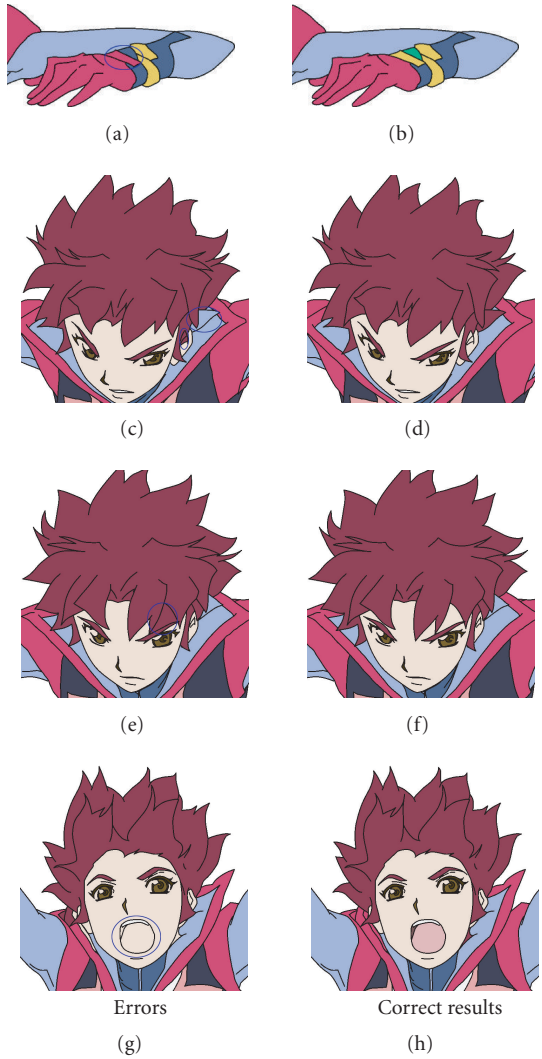


Figure 10: Error Analysis for Test 1.

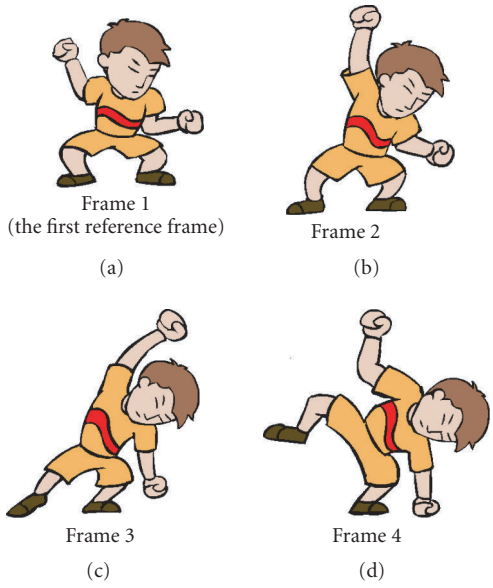


Figure 11: Test 2 (resolution: 550 × 400 pixels).

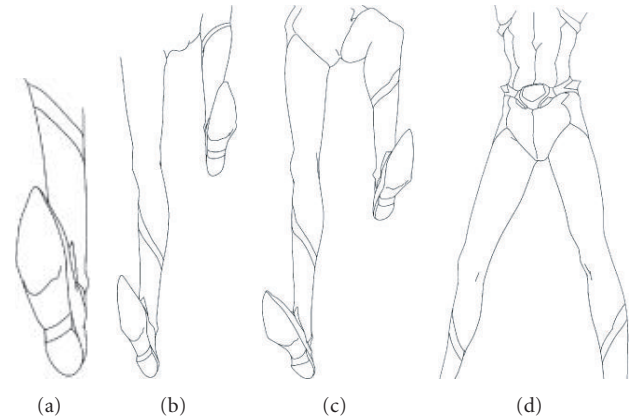


Figure 12: Other frames in the cut used in Test 1 (courtesy of Anime International Co., Inc., Japan).

5. CONCLUSION AND FUTURE WORK

In this paper, our previous work on character registration is enhanced with a refined skeleton extraction procedure and a novel character relocation method. With the enhanced character registration, occlusion can be correctly detected and located in components segmented from the character. Subsequently, two different matching methods are applied to regions in components without and with occlusion, respectively. A practical animation sequence from a cut of "Justeen" animation trailer is applied to validate the approach. A high coloring accuracy is achieved. It shows that the proposed auto coloring approach with enhanced character registration is applicable to practical animation production.

Nevertheless, there are limitations. The proposed approach requires that each frame has a stable branch, which is not always true among a long sequence. For example, the cut we used in Test 1 contains 12 frames in total, and the first 4 frames are illustrated in Figure 12. Due to the information loss, no stable branch exists in these frames. So they cannot be automatically colored using our approach. Future research work will be focusing on the auto coloring for these kinds of special cases and new relocation methods.

ACKNOWLEDGMENT

This work has been partially supported by the Science and Engineering Research Council (SERC) Grant no. 052 015 0024 Ref: 44, which is awarded by the Agency for Science and Technology Research (A∗STAR) and administered through the Singapore National Grid Office.

REFERENCES

[1] J. Qiu, H. S. Seah, F. Tian, Z. Wu, and Q. Chen, "Feature- and region-based auto painting for 2D animation," *The Visual Computer*, vol. 21, no. 11, pp. 928–944, 2005.

[2] J. Qiu, H. S. Seah, F. Tian, Q. Chen, Z. Wu, and M. Konstantin, "Auto coloring with character registration," in *Proceedings of the International Conference on Game Research and Development*

(CyberGames '06), vol. 223, pp. 25–32, Perth, Australia, December 2006.

[3] N. D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing hierarchical curve-skeletons of 3D objects," *The Visual Computer*, vol. 21, no. 11, pp. 945–955, 2005.

[4] P. Golland and W. E. L. Grimson, "Fixed topology skeletons," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 10–17, Hilton Head Island, SC, USA, June 2000.

[5] J. Qiu, H. S. Seah, F. Tian, Q. Chen, and K. Melikhov, "Topology enhanced component segmentation for 2D animation character," in *Proceedings of International Workshop on Advanced Imaging Technology*, pp. 30–35, Okinawa, Japan, January 2006.

[6] S. M. Smith, "Edge thinning used in the SUSAN edge detector," Internal Technical Report TR95SMS5, Defence Research Agency, Surrey, UK, 1995.

[7] C. Patmore, *The Complete Animation Course: The Principles, Practice and Techniques of Successful Animation*, Thames & Hudson, London, UK, 2003.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, Mass, USA, 1992.

[9] J. Qiu, H. S. Seah, F. Tian, Q. Chen, and Z. Wu, "Enhanced auto coloring with hierarchical region matching," *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 463–473, 2005.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 834616, 6 pages
doi:10.1155/2008/834616

Research Article

Strategic Team AI Path Plans: Probabilistic Pathfinding

Tng C. H. John,[1] Edmond C. Prakash,[2] and Narendra S. Chaudhari[1]

[1] *School of Computer Engineering, Nanyang Technological University, Singapore 639798*
[2] *Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, UK*

Correspondence should be addressed to Edmond C. Prakash, e.prakash@mmu.ac.uk

Received 29 September 2007; Accepted 13 December 2007

Recommended by Kok Wai Wong

This paper proposes a novel method to generate strategic team AI pathfinding plans for computer games and simulations using probabilistic pathfinding. This method is inspired by genetic algorithms (Russell and Norvig, 2002), in that, a fitness function is used to test the quality of the path plans. The method generates high-quality path plans by eliminating the low-quality ones. The path plans are generated by probabilistic pathfinding, and the elimination is done by a fitness test of the path plans. This path plan generation method has the ability to generate variation or different high-quality paths, which is desired for games to increase replay values. This work is an extension of our earlier work on team AI: probabilistic pathfinding (John et al., 2006). We explore ways to combine probabilistic pathfinding and genetic algorithm to create a new method to generate strategic team AI pathfinding plans.

1. INTRODUCTION

A popular game with heavy team AI is "Full Spectrum Warrior" (FSW), a console game on Xbox. This game is a downsized version of "Full Spectrum Command" on the PC platform. The game "Full Spectrum Command" is actually a simulation of the real-world behavior of the US Army. This game was originally used in the military for leadership training as well as decision making training. The game includes real-world army movements such as "bounding" and ducking. The main feature of the game is its team AI. The team AI of the game is actually derived from the real world and simulates how a real person will behave. The purpose of this game is to command two teams of soldiers to accomplish a mission.

Even though it is a great game, one area it can improve on is the team AI plan. Players may feel that opponents always appear at the same places after playing repeatedly, as most team AI plans use A* algorithm or look up tables as their main pathfinding techniques. These techniques always produce the same path if the source and destination locations are the same. On the other hand, Probabilistic is able to produce variations to the path even if the source and destination locations are the same.

Replay value can easily be added to the game by creating variations to the opponent team plans. When the opponents have different plans, they move differently, thus the players cannot always predict the opponents' locations. This work uses probabilistic pathfinding algorithm to obtain variations of team AI path plans.

Section 2 talks about related work, existing problem, and a scenario that strategic team AI path planning can be applied. Section 3 gives an introduction about team AI and probabilistic pathfinding. It explains how team AI is created in [1] by combining different systems. Section 4 describes the main method of strategic team AI path plan generation. Section 5 suggests a method that can be used to optimize this work. Section 6 shows some interesting strategies generated by this team AI path plan generation method. The final section, Section 7, comprises conclusion, other applications, and future suggestions for this work.

2. RELATED WORK

In this paper, team AI pathfinding and team AI plans are the same. Team AI pathfinding refers to the different paths taken by teammates to reach a desired place. An example is a team of enemies enter a room from different doors to trap and capture the player. Team AI pathfinding plans are popular in computer games. With good team plan, the game difficulty level increases, making the game more challenging and helps to showcase intelligent behavior of the game.

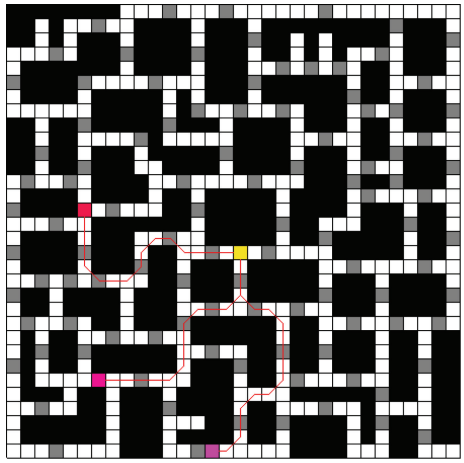


Figure 1: Situation without team AI.

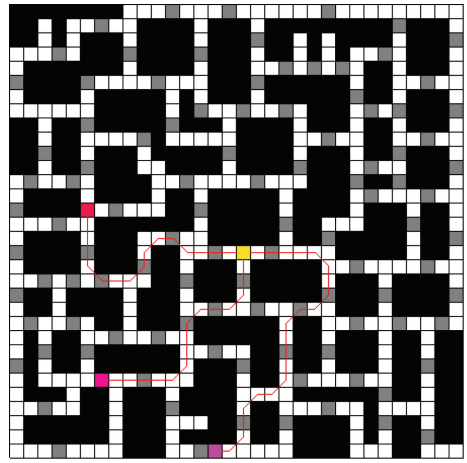


Figure 2: Situation with team AI.

In this work, we use and extend our earlier work on probabilistic A* pathfinding algorithm [1]. Further readings on A* pathfinding can be found in [2–4]. Bourg and Seeman [4] provide other data representation of the A* pathfinding. A very useful article by Pinter [5] discusses methods to modify a raw path generated from a path search to form a more convincing traveling path.

An interesting work by Kamphuis et al. [6] attempts to simulate tactical pathfinding in urban environments for a small group of characters for games and simulations. The characters use tactical information such as road maps and special locations for pathfinding. This work uses common A* data structure and a list of gateway points. The tactical pathfinding [6] algorithm is able to run in real time with the help of a preprocessing step. For this work to run in real time, no preprocessing is needed. With optimization, it can even run more efficiently.

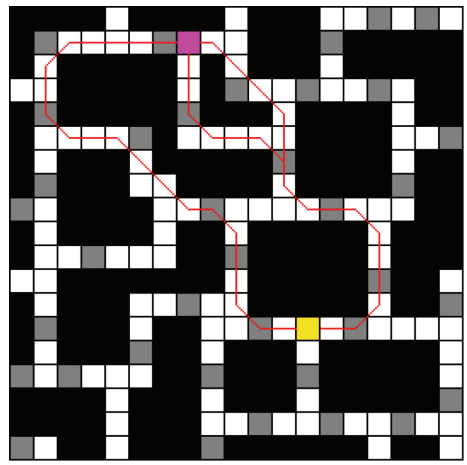


Figure 3: Different paths generated by probabilistic pathfinding.

3. PREVIOUS WORK

This section is an introduction to team AI: Probabilistic pathfinding. The detail implementation and algorithm can be found in [1]. Team AI can be shown to exist if teammates coordinate to trap or capture an opposite team. Figure 1 shows a situation when team AI is not enabled. The enemy teammates find the shortest path to capture the player.

Figure 2 shows a situation with team AI enabled. The enemy teammates surround, trap, and capture the player.

Probabilistic is a modified version of A* algorithm with an addition ability to generate different paths controlled by a probability variable. The variable controls how different the paths differ from the shortest path. The paths may not be the shortest, but they are one of the shorter paths. In general, if the variable is 0, then probabilistic pathfinding will behave exactly the same as a usual A* pathfinding. If the variable is set to 1, it will always produce a different path that is not the shortest. For more details refer to [1]. Figure 3 below shows an example of an enemy character following different paths to pursuit the player character.

Notice that there are light grey squares in Figures 1, 2, and 3. They are actually the gateways of the map. Gateway is narrow path in the environment that opens up to a bigger path before and after the gateway. That is, a gateway is a narrow link between two spaces. Figure 4 illustrates a gateway.

A blackboard messaging system is developed in [1] to facilitate communications between characters of a team. In short, blackboard is a place for characters to "write" useful information and let other teammates read it. After every teammate read, the message is deleted by the message writer. Figure 5 shows the concept of a blackboard.

Putting together probabilistic pathfinding, gateways information, and message system, we achieved what is shown in Figure 2, the complete working team AI in [1].

4. PATH PLAN GENERATION

In this section, we illustrate the method used to generate high quality team AI path plans. The main idea is to test the team AI path plans with a fitness function. The fitness will

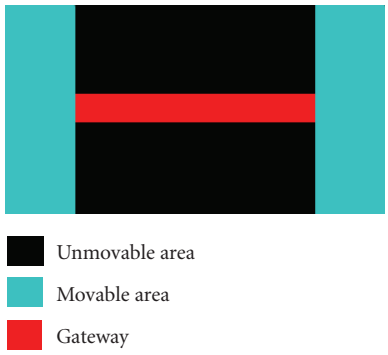


Figure 4: Illustration of a gateway.

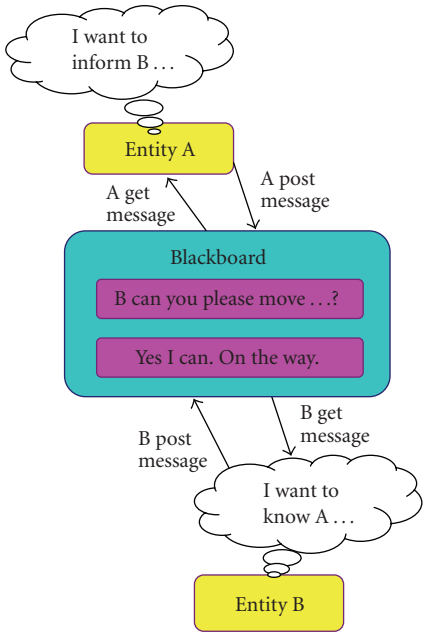


Figure 5: Different paths generated by probabilistic pathfinding.

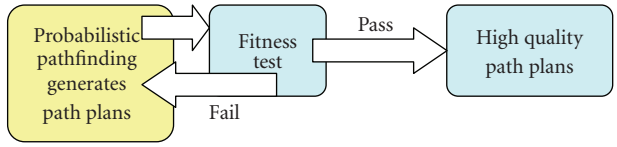


Figure 6: Path plan generation flow chart.

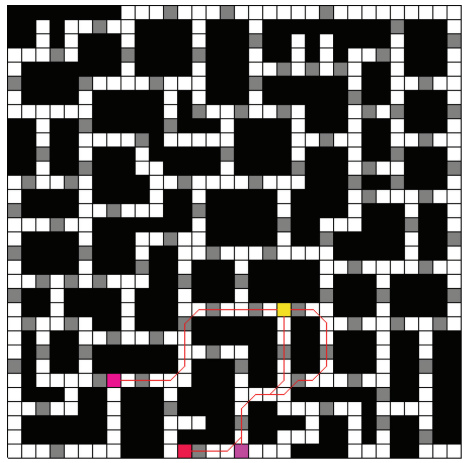


Figure 7: Path with overlapping regions.

determine whether the team plan is good enough. If the quality is bad, a new team AI path plan search will be conducted. Figure 6 shows the flow chart for path plan generation.

This idea is mainly inspired by genetic algorithms [7], where a fitness function is used to test the quality of the genes combination. The genes combination is formed based on its parents, some manipulation, and some randomness. The better the quality of the combination of genes is, the higher chance it will survive. The bad quality gene combinations get eliminated. Team AI path plan works the same way. Different path plans are generated by probabilistic pathfinding. Treat each path plan as a combination of genes. A fitness function is used to test path plan. If it is not good enough, it will be eliminated. A new search will be conducted. The cycle repeats until a satisfactory quality path plan is obtained.

A fitness test can be a simple function that calculates the distance between two characters. For example in a game, it is not acceptable for two teammates to get too near to each other or their paths overlap. An alternative could be for the team to explore a bigger area of the map to gain resources and familiarize with the terrain. It all depends on the game play. Therefore, the fitness test for such a path plan will fail if two teammates get too close to each other. Fitness test can also be constraints of the team path plan.

Figure 7 shows a team path plans that are not acceptable because of overlap paths.

The path of the bottom-right enemy character follows the shortest path to the player. With team AI enabled, that bottom two characters have to trap the player through different entrances. However, due to overlap paths constraint, the path plan is discarded. A new path plan is conducted and shown in Figure 8. By comparing the plan shown in Figure 7 and Figure 8, the team path plan in Figure 8 is better than that of Figure 7 according to overlap constrains.

The following are three ways to test the team paths with the fitness function. They are illustrated below.

4.1. *Iterative test*

The fitness test is conducted after the whole team found its path. If the fitness test fails, a new path plan search will be conducted. However, the old path plan is saved. This is to prevent the system from doing too many searches and slow down the game. The user can specify a number of maximum searches to perform. If the maximum number of searches is reached, the path plan with the highest fitness will be selected. The user may also choose to terminate the

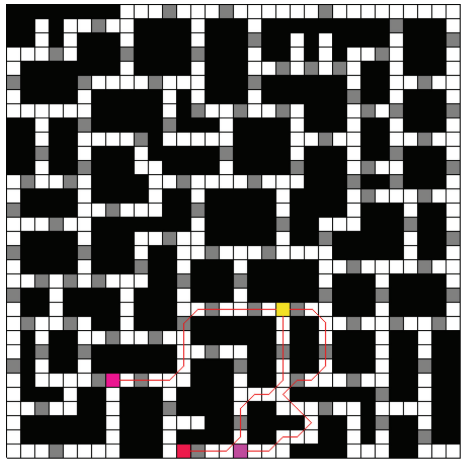


Figure 8: Path without overlap.

```
Path QualityPathPlanForWholeTeam() {
    Path pathOfWholeTeam;
    do {
        pathOfWholeTeam = null;
        While(TeamPathPlanNotComplete()) {
            pathOfWholeTeam + =
                ProbabilisticPathForOne();
            }
    } while(FitnessTest(pathOfWholeTeam)==fail);
  Return pathOfWholeTeam;
}
```

Algorithm 1

search once a path plan passed the first fitness test. This method is efficient if the fitness test seldom fails on path plans. This method is the easiest and fastest to implement. No modification is needed for probabilistic pathfinding generation algorithm. No modification is needed for team path planning. The exact algorithm is already shown in Figure 6. Algorithm 1 shows the pseudo code algorithm.

4.2. Step test

Algorithm 2 shows the pseudo code of the step test.

The fitness test is conducted at every segment of the path. This is the extreme opposite end of iterative test. On selection of the next node (using probability pathfinding search), if it does not pass the fitness test, another node will be chosen instead. This method is good if iterative test always fails and the number of characters is small. As opposed to iterative test, modification is needed for probabilistic pathfinding generation algorithm. The fitness test function has to be included into the probabilistic pathfinding algorithm.

4.3. Progressive test

The test will be conducted after each character found its path. This is a middle solution between the iterative test

```
Path QualityPathPlanForWholeTeam () {
    Path pathOfWholeTeamSoFar = null;
    While (TeamPathPlanNotComplete ()) {
            pathOfWholeTeamSoFar + =
    ProbabilisticPathForOne (pathOfWholeTeamSoFar);
      }
      Return pathOfWholeTeamSoFar;
}
Path ProbabilisticPathForOne (Path
pathOfWholeTeamSoFar) {
      Path currentMemberPath = null;
      Path temp = null;
      do {
            currentMemberPath += selectANextNode ();
            do {
                  temp = pathOfWholeTeamSoFar +
currentMemberPath.selectADifferentNode ();
            } while (FitnessTest(temp) == fail);
    } while
(currentMemberPath.SearchNotComplete ())
    return currentMemberPath;
}
```

Algorithm 2

```
Path QualityPathPlanForWholeTeam () {
    Path pathOfWholeTeamSoFar = null;
    Path temp = null;
    Path memberPath = null;
    do {
        do {
            memberPath = ProbabilisticPathForOne ();
          temp = pathOfWholeTeamSoFar + memberPath;
          } while (FitnessTest(temp) == fail)
          pathOfWholeTeamSoFar += memberPath;
      } while (TeamPathPlanNotComplete ());
      Return pathOfWholeTeamSoFar;
}
```

Algorithm 3

and the step test Table 1. It is based on each character. After each character has found its path, the fitness test will be performed. If the test fails, the character will choose another path. This is the best method if the iterative test and step test always fail. Modification needs to be made to team path planning. Fitness test is conducted per character (see Algorithm 3).

5. OPTIMIZATION

Fitness testing should be cheap if it does not involve calculation of huge set of constraints and variables. The load of this system comes from A* pathfinding. This means that all optimization techniques applicable to A* pathfinding are here. A common optimization technique for A* pathfinding

Table 1: Summary fitness function test.

Methods	Advantage	Best for cases
Iterative test	Easiest and fastest to implement	Many high quality solutions Fitness test mostly passes
Progressive test	A general solution that can solve most cases	Average solution In the middle of iterative test and step test
Step test	Guaranteed to have a solution if it exists	Few high quality solutions Fitness test mostly fails

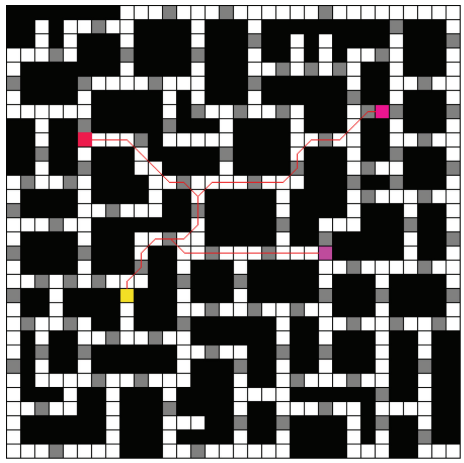


Figure 9: Combine force strategy.

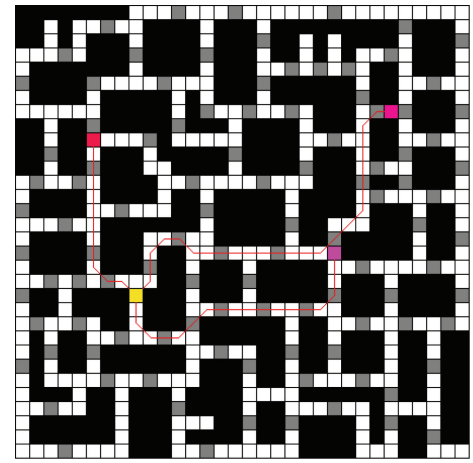


Figure 10: Trap strategy.

is search by parts so that the A* search execution is spread out over many frames.

6. ANALYSIS OF STRATEGIES

One useful feature of this team AI path plan generation method is to generate various useful strategies. These strategies can be applied to first person shooting team games as AI opponents or enemies. With such interesting strategies, the replay value of the game increases. The difficulty level increases and it becomes more challenging for player to defeat the enemies. This section analyzes interesting strategies generated.

Figure 9 shows the combined force strategy. In this particular strategy, it is the reverse. Joining teammates together synergized the power of the enemy team and increased the chance of success for eliminating the player. From Figure 9, the enemies join forces along the way and attack the player together in a single path. The fitness function to such a plan is to test the path before the destination (the player position) and ensure that before the destination all three teammates must be together. This is the first constraint. This fitness function will eliminate plans that do not combine forces before they encounter the player. The second constraint controls can be how early the teammates must combine their forces before they encounter. The earlier they meet the higher chance of success in their mission. For this example, the second constrained fitness function is not tested. As long they are able to meet before encountering the player, it is a good strategy.

Figure 10 shows a trap strategy. It is the reverse strategy of Figure 9. In this case, the team may have higher chance of success for killing the player. The trap strategy aims to trap the player at different directions. As far as possible, this means that the enemy teammates should not have overlap paths. So there are two constrained fitness functions for this example. The first constraint means, as far as possible, that the teammates must not encounter the player in the same direction. In general, use the pigeonhole theorem [8]. Number of teammates in same direction cannot be greater than teammates divided by number of directions. This is to ensure equal distribution of teammates in each direction. The second constraint is to avoid crossing paths of the teammates. This is to create higher chance of search space if the player escapes somewhere else.

Figure 11 shows a similar strategy as in Figure 10. However, a third fitness constraint is applied. The third constraint is the minimum distance away from the path of teammates. This is the *explore and trap strategy* . No doubt, the main objective of the teammates is to trap and capture the player in different directions and paths. In addition, the enemy teammates have a secondary objective to explore a wider area of the map. This will facilitate their future plans, actions, or operations. From Figures 10 and 11, with an additional fitness function, the path generated is different. A secondary objective can be included with additional constraints.

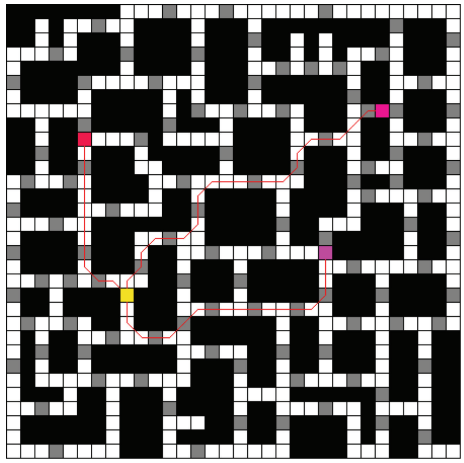


Figure 11: Explore and trap strategy.

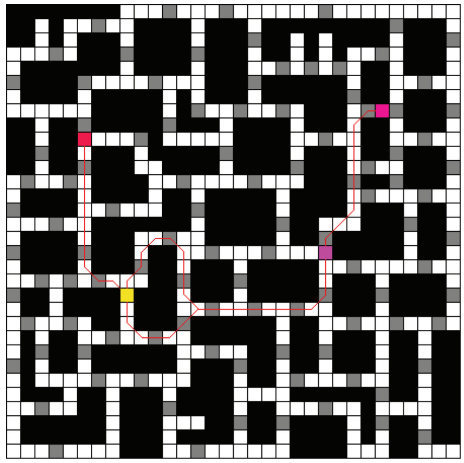


Figure 12: Combine and split trap strategy.

Figure 12 shows the most interesting strategy. One of the teammates follows the path of another teammate. To make analysis easier, the enemy teammate right of the player is known as $E1$ and the enemy teammate on top of $E1$ will be $E2$. In this situation, the map can be full of land mines. For every grid path that a character moves, it needs to remove all the landmines and make sure it is safe to travel before it can proceed to the next grid. $E1$ is ahead of $E2$ to the player. $E2$ tries to follow the path of $E1$ because in that case, $E2$ does not need to waste effort removing all landmines. This is the first constrained fitness function, which is try to overlap paths if they are along the way.

This is also a trap strategy because all teammates trap the player from different directions. This should be the second constraints. That is, all teammates should try to trap the player in different directions.

Using these two constrained fitness functions, a very nice strategy is generated from the team AI path plan method. The teammates know what they are doing. They work together and save effort for removing landmines. When they are near to the player, they split their ways to trap the player.

7. CONCLUSION

This paper proposes a new method inspired by genetic algorithm to generate interesting and high quality path plans for team AI. Probabilistic pathfinding is used for path search and blackboard architecture is used for communication between teammates. The path generation algorithm runs in real time without any preprocessing. Only the standard graph data and a list of gateway points are needed for the A* search. Controllable randomness allows the path generation to be tuned easily. The dynamic generation of path plans adds replay value to games. In addition, interesting path plans generated from this method are able to showcase the AI intelligences of the game which is a good selling point for games.

This strategic path plan generation method can apply to other applications that involve path searching. It is best for applications that have many solutions. A good example is traffic control system or GPS system. Such systems can plan the path of vehicles to avoid congestion. Congestion condition is used as constraints for fitness test to fail. For example, a congestion condition can be that the number of vehicle traveling along a road must be less than a maximum number. Another good application is goal planning with many different ways of achieving the goals. This path plan generation method can generate good quality path plans to achieve the goals. A real example is a mission-based game where there are many ways to solve a mission. Choosing a good plan to solve the mission can bring out the intelligence of game characters.

The path plan generated using this method is by trial and error. Generate a path, test it, and discard it if it is not good enough. A future step to go from here is to generate path plans by functions or heuristic. An example is to add a fitness function as a heuristic function to the probability pathfinding algorithm. With the fitness heuristics function, the path plans generated will always be good. This will prevent all the wasteful discards of low quality path plans.

REFERENCES

[1] T. C. H. John, E. C. Prakash, and N. S. Chaudhari, "Team AI: probabilistic pathfinding," in *Proceedings of the International Conference on Game Research and Development*, vol. 223 of *ACM International Conference Proceeding*, pp. 191–198, Perth, Australia, December 2006.

[2] S. Rabin, *AI Game Programming Wisdom 2*, Charles River Media, Hingham, Mass, USA, 2004.

[3] M. Buckland, *Programming Game AI by Example*, Wordware, Plano, Tex, USA, 2005.

[4] D. M. Bourg and G. Seeman, *AI for Game Developers*, O'Reilly, Sebastopol, Calif, USA, 2004.

[5] M. Pinter, "Gamasutra," http://www.gamasutra.com/features/20010314/pinter_01.htm.

[6] A. Kamphuis, M. Rook, and M. H. Overmars, "Tactical path finding in urban environments," 2005, http://www.cs.uu.nl/centers/give/movie/index.php.

[7] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ, USA, 2002.

[8] D. B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2000.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 873913, 11 pages
doi:10.1155/2008/873913

Research Article

Hierarchical Pathfinding and AI-Based Learning Approach in Strategy Game Design

Le Minh Duc, Amandeep Singh Sidhu, and Narendra S. Chaudhari

School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore, Singapore 639798

Correspondence should be addressed to Le Minh Duc, minhducle@pmail.ntu.edu.sg

Received 10 October 2007; Accepted 26 February 2008

Recommended by Kok Wai Wong

Strategy game and simulation application are an exciting area with many opportunities for study and research. Currently most of the existing games and simulations apply hard coded rules so the intelligence of the computer generated forces is limited. After some time, player gets used to the simulation making it less attractive and challenging. It is also costly and tedious to incorporate new rules for an existing game. The main motivation behind this research project is to improve the quality of artificial intelligence- (AI-) based on various techniques such as *qualitative spatial reasoning* (Forbus et al., 2002), *near-optimal hierarchical pathfinding* (HPA*) (Botea et al., 2004), *and reinforcement learning* (RL) (Sutton and Barto, 1998).

1. INTRODUCTION

Although strategy games have been around for above ten years, AI is still the biggest challenge in games with many unsolved problems. In this research, RL is chosen to further develop AI techniques. RL is learning from interaction with an environment, from the consequences of action, rather than from explicit teaching. In order to apply RL successfully, some of qualitative spatial reasoning techniques and HPA* are employed to design a better framework. In addition, real-time strategy (RTS) genre is selected for implementing the game to demonstrate the result.

Here are the milestones of this research work. Firstly, a game idea is brainstormed and implemented into a complete game demo called StrikeXpress with all the basic characteristics of a RTS. Secondly, the game demo is optimized with more expressive spatial representations, better communication of intent, better pathfinding, and reusable strategy libraries [1]. Finally, the RL is applied to the game's AI module. The paper is organized as follows. In Section 2, we review important concepts used in this project and briefly outline the development platform and tools used to create the game demo. In Section 3, we discuss the approaches for pathfinding and qualitative spatial reasoning techniques. We describe the framework for RL in Section 4 and conclude in Section 5.

2. LITERATURE REVIEW

2.1. Game design process

Game is made of many components, and game design process has to go through many steps as discussed in detail in [2]. However, making a complete commercial game is not our intention; our main focus is to build a basic game to demonstrate the research idea. For this purpose, we follow a simple game design process as shown in Figure 1. In *Concept phase*, we have to brainstorm the game story, look for concept arts, and choose the development platform. *Design phase* is mainly to design models, game levels based on design documents. *Components Implementation phase* is to implement components such as user interface, visual and audio effects, game mechanics, and AI. The next steps are *integration*, *fine tuning, and testing* before launching the game demo. The most challenging issue is how to implement machine learning feature nicely without affecting the game flow. Figure 3 shows the overall project architecture where the left part is game design process, and the right part is the framework for RL.

2.2. Qualitative spatial reasoning

Qualitative representations carve up continuous properties into conceptually meaningful units [3]. Qualitative spatial

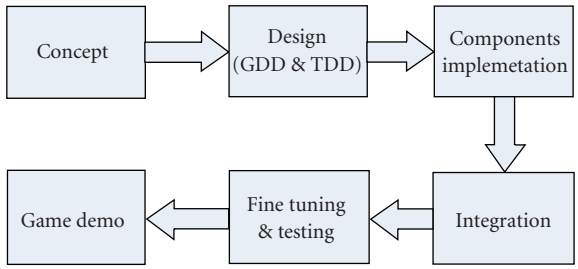


Figure 1: Game design process.

representations carve space into regions [4], based on a combination of physical and task-specific constraints.These techniques can provide a more humanlike representation of space and can help overcome the difficulties of spatial reasoning. This will let us build strategy AIs that more effectively use terrain to achieve their goals, take orders, and find their way around. Moreover, decoupling the spatial representation in the AI from the spatial implementation in the game engine constitutes a large step toward making reusable AIs, driving down development costs while improving AIs' subtlety [1]. The approach is discussed in detail in Section 3 together with pathfinding.

2.3. Near-optimal hierarchical pathfinding

The popular solution for pathfinding is A* algorithm. However, as A* plans ahead of time, the computational effort and size of the search space required to find a path increase sharply. Hence, pathfinding on large maps can result in serious performance bottlenecks. Therefore, HPA* [5] is used to overcome the limitations of A*. The main idea is to divide and conquer—break down a large task into smaller specific subtasks. HPA* will be discussed in detail in Section 3.

2.4. Hierarchal AI-based learning

In Figure 2, we use simple American hierarchal military structure to demonstrate the idea. An Army Lieutenant typically leads a platoon-size element (16 to 44 soldiers) to perform specific tasks given by higher commissioned officer such as Captain, Major, Colonel, or General. Similarly in the game, platoon represents *the lowest level agents* (LLA-) which perform real actions like move, run, shoot, and guard. Lieutenant represents *the middle level agent* (MLA) which decides the best strategy for the platoon such as to find the optimal paths, split the platoon into subgroups to move on different paths, decide the suitable time for engagement, retreat or call for reinforcement, and report results to higher officer. Higher commissioned officer represents *the highest level agent* (HLA) in the game that uses RL to learn from the environment and the consequences of actions performed by lower level agents to decide next actions such as to send forces to engage the enemy at coordinate (x, y, z), to agree or

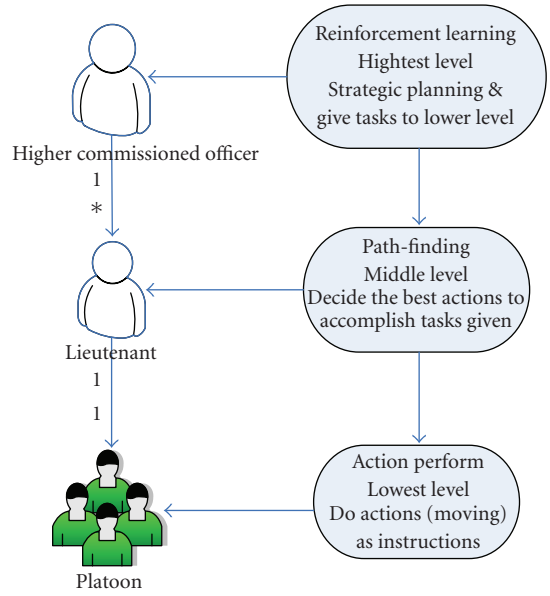


Figure 2: AI-based learning structure.

reject to send reinforcement, and to set up a series of strategic actions.

We notice that LLA is the easiest to implement as most of the actions are primitive and can be taken care of them by the game engine's built-in functions. HLA can be realized based on proven and established RL algorithm, provided that there is sufficient information for decision making—as shown in Figure 3; machine learning structure is already designed for RL. It is realized that the most difficult and bottleneck part is MLA where the game can be slow down noticeably, or the AI can become stupid due to improper, nonoptimized pathfinding, and data structure. Besides, most of the strategic actions planed by HLA involve some kind of movement. Without an efficient MLA, RL may not work properly.

2.5. Development tools

The game engine used to create game demo, StrikeXpress, is 3D Gamestudio (http://www.3dgamestudio.com/). MySQL is used for database storage (http://www.mysql.com/), and Matlab is used for running RL function (http://www.mathworks.com/products/matlab/). In addition, to connect between these tools, we must use some DLL extensions supported by 3D Gamestudio which is basically an external library that adds functions to the game engine. A piece of code written in DLL form may run faster than that written in game scripting language due to its precompilation. Theoretically everything, MP3 or MOD players, a physics engine, another 3D engine, or even another scripting language, can be added to the engine this way. Therefore, in order to make connection between the game engine and MySQL, we use a DLL extension called A6mySQL which is written in C++ (http://www.plants4games.com/hmpsplit/files/A6MySQL_Public_Release.rar).

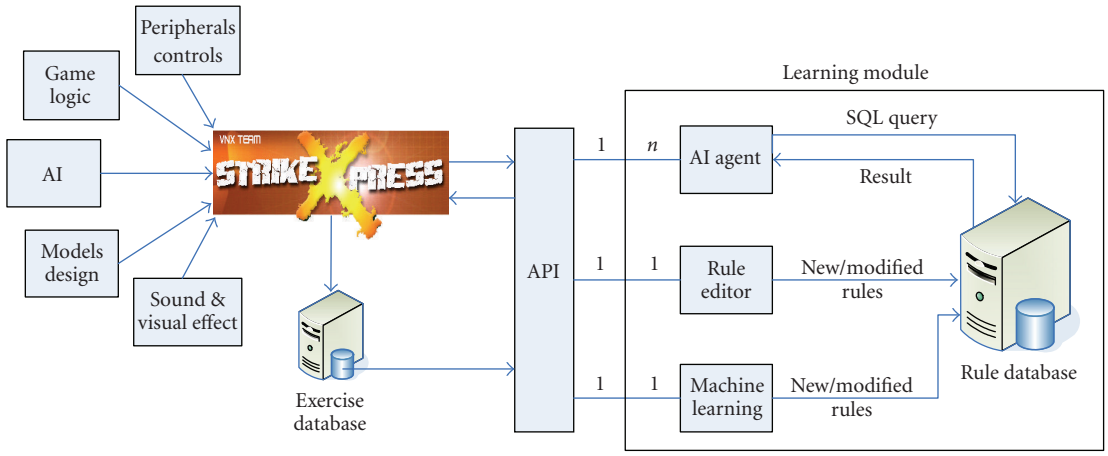


Figure 3: The overall project architecture.

To make connection between MySQL and Matlab, the plugin created by Robert Almgren is used (http://mmf.utoronto.ca/resrchres/mysql/).

3. PATHFINDING APPROACHES

Pathfinding on RTS games is complex because the environment is dynamic; there are lots of units which continuously move around the map and its scope equals the size of the level. This section is to demonstrate the use of two pathfinding approaches in the game: *Points of visibility* [6] and HPA*.

3.1. *Points of visibility*

Points of Visibility algorithm uses waypoints scattered around the level. A set of waypoints are connected together to create a network of movement directions. As shown in Figure 4, this network alone is sufficiently enough to guide a unit to transverse every obvious location of the map. In this approach, for simplicity, all the waypoints are placed manually, but the connections between those waypoints are done automatically like in Figure 5. How the waypoint is placed will make or break the underlying pathfinding code. The idea is to build a connected graph which will visit all places of our level. In human architecture, particularly tight corridors and other areas where the environment constrains the agents' movement into straight lines, waypoints should be placed in the middle of rooms and hallways, away from corners to avoid the wall-hugging issues. However, in large rooms and open terrain, waypoints should be placed at the corners of obstacles in a game world with edges between them. It will help generate paths almost identical to the optimal one.

In the graph making process, we select one entity to be responsible for creating and loading the graph data file. What the graph making process actually does is to let the selected entity move from one waypoint to another. A waypoint A is said to be connected with waypoint B if the selected entity is able to walk from A to B in straight line. The size of this selected entity will be considered when creating the graph so we must choose it wisely. After the graph making process is completed, all the connections will be stored in a data file. In the game, when pathfinding is invoked, it will process on the graph loaded from the file, and we can use any search algorithms to find the way. In this project, for simplicity, we deploy Dijkstra search algorithm.

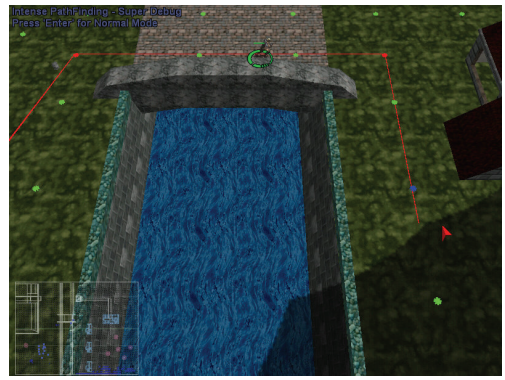


Figure 4: A path shown in Debug mode.

Advantages

Points of Visibility are being used today by more than 60% of modern games. It is simple and efficient thank to node-based structure. It is particularly useful when the number of obstacles is relatively small and they have a convex polygonal shape. When encountering slopes, hills, and stairs, we will get better results if placing a waypoint every short distance to fully cover it. Also, the graph does not need to be fully connected. The algorithm can handle the case where a level is split into two parts, and the player teleports from one part to the other. We can apply any search algorithm to this approach. We also can smooth the paths to make it look

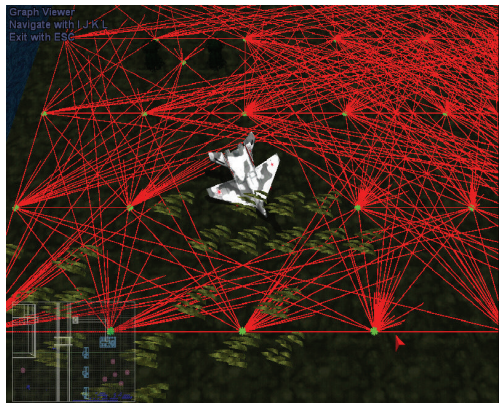


Figure 5: Waypoints placed manually and connections generated automatically.

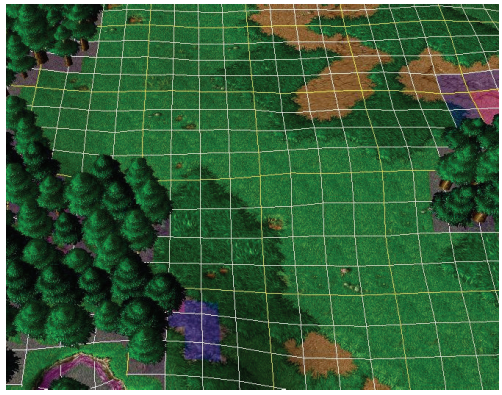


Figure 6: Transform terrain to large grid [screenshot taken from Warcraft III map editor].

more natural. The graph making process is also useful for debugging as all the waypoints and connections are displayed explicitly like Figure 5.

Disadvantages

The efficiency of the method decreases when many obstacles are present and/or their shapes are not a convex polygon or the level is open terrain with dense collection of small size obstacles. Modeling such a topology with this approach would result in a large graph with short edges. Therefore, the key idea of traveling long distances in a single step would not be efficiently exploited. The need for algorithmic or designer assistance to create the graph is also troublesome. In addition, the movement needs a lot of adjustment to be realistic, and the complexity increases fast when dealing with multiagents.

3.2. The HPA* algorithm

This technique is highly recommended based on its efficiency and flexibility to handle both random and real-game maps with a dynamically changing environment using no domain-specific knowledge. It is also simple and easy to implement. If desired, more sophisticated, domain-specific algorithms can be plugged in to increase the performance.

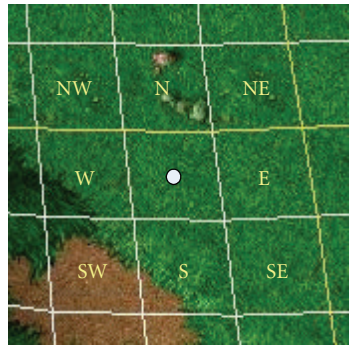


Figure 7: Representation of the first 8 neighbor cells.

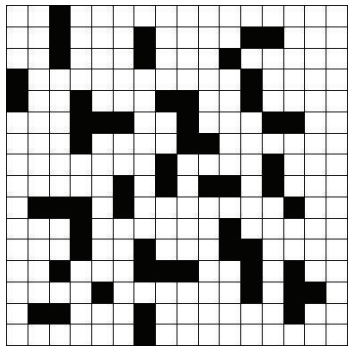


Figure 8: More expressive representation level grid.

3.2.1. HPA* preprocessing phase (offline)

Transform the level to large grid

The entire level is transformed into large grid with equal cells' size as shown in Figure 6. All the cells will be scanned. From all the accessible cells, we will check the height and any special values of each cell to determine its cost to use in A* algorithm. Hence, each cell can be treated as a node similar to waypoint in previous algorithm. All the cells' information will be put into an array for further processing.

Prelink the cell array

After transforming the level to large grid, we scan through each cell to see what surrounding cells can actually link to (NE, N, NW, E, W, SE, S, SW) as shown in Figure 7. For a surface with many cliffs, a cell on a cliff may not be reachable from its neighbor if the slope is too great. As a result, the level is transformed from the original in Figure 6 to more expressive representation grid like in Figure 8 where the black cell is totally inaccessible and white cell is accessible by some of its neighbors. The cost of each white cell may be different.

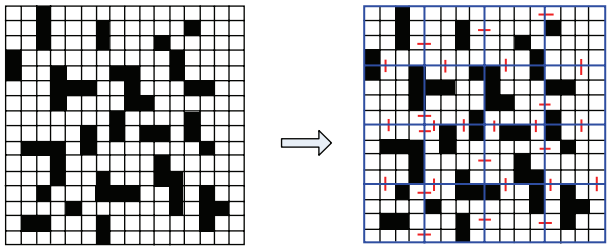


Figure 9: Grid to 16 subgrids.

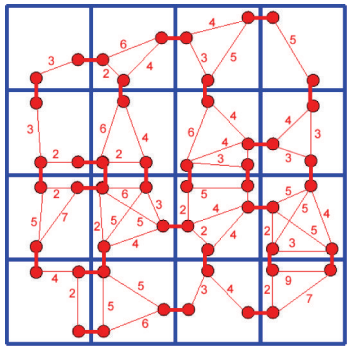


Figure 10: Abstract subgrid connectivity graph.

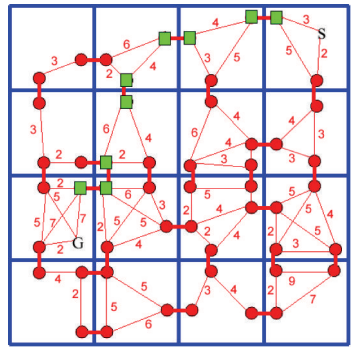


Figure 11: Use A* to find path from S to G with cost 29.

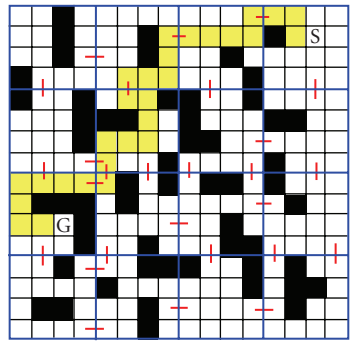


Figure 12: Path refinement with cost 29.

Divide a large grid into smaller clusters and find entrances between these clusters

The grid in Figure 8 can be divided into subgrids (clusters) in many ways, as shown in Figure 9. An entrance is a maximal obstacle-free segment along the common border of two adjacent clusters c1 and c2 [5]. Entrances are obtained for each subgrid in the same manner as larger grid and the red lines connect the resulting entrance nodes.

Build abstract subgrid connectivity graph

Transitions are used to build the abstract problem graph. For each transition, we define two nodes in the abstract graph and an edge that links them. The edge represents a transition between two clusters is called interedge. Each pair of nodes inside a cluster is linked by an edge called intraedge. The length of an intraedge is computed by searching for an optimal path inside the cluster area. We only cache distances between nodes and discard the actual optimal paths corresponding to these distances. If desired, the paths can also be stored, for the price of more memory usage [5]. After building the abstract graph like Figure 10, this graph is saved into a precompiled node list file for that level.

3.2.2. Pathfinding phase (online)

Add S and G to abstract graph and use A search*

When the game is loaded, we will also load its precompiled node list. The first phase of the online search connects the starting position S to the border of the cluster containing S by temporarily inserting S into the abstract graph. Similarly, connecting the goal position G to its cluster border is handled by inserting G into the abstract graph. After S and G have been added, A* is used to search for a path between S and G in the abstract graph. This is the most important part of the online search where heapsort and heap structure are used. It provides an abstract path, the actual moves from S to the border of S's cluster, the abstract path to G's cluster, and the actual moves from the border of G's cluster to G [5] as shown in Figure 11. In case S and G change for each new search, the cost of inserting S and G is added to the total cost of finding a solution. After a path is found, we remove S and G from the graph. Consider the case when many units have to find a path to the same goal, we insert G once and reuse it. If among these units there are some units close to each other, this group of units can share the same search operation. In the case the destination can be reached without obstacles in the way, a simple linear path should be chosen instead. The cost of inserting G is amortized over several searches. In general, a cache can be used to store connection information for popular start and goal nodes.

Refine path as needed

Path refinement translates an abstract path into a low-level path. Each cluster crossing in the abstract path is replaced by an equivalent sequence of low-level moves as shown in Figure 12. If the cluster preprocessing cached these move

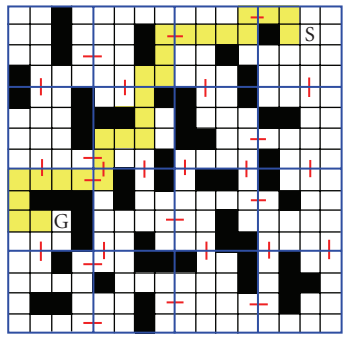
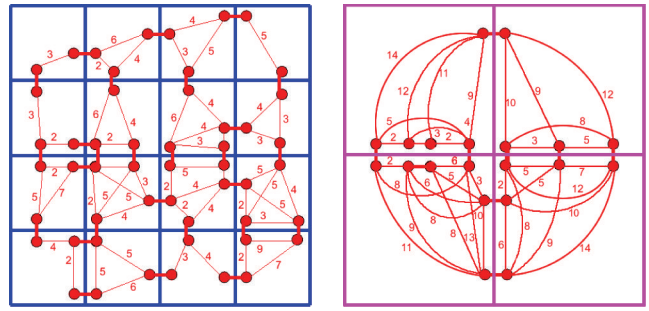


Figure 13: Path smoothing with cost 27.



(a) (b)

Figure 14: Multi-level abstract graphs with 16 "1-clusters" and 4 "2-clusters".

sequences attached to the intraedges, then refinement is simply a table look-up. Otherwise, we perform small searches (using A*) inside each cluster along the abstract path to rediscover the optimal local paths. Consider a domain where dynamic changes occur frequently, after finding an abstract path, we can refine it gradually as the character navigates toward the goal. If the current abstract path becomes invalid, the agent discards it and searches for another abstract path. There is no need to refine the whole abstract path in advance [5].

Apply smoothing

The topological abstraction phase defines only one transition point per entrance and gives up the optimality of the computed solutions. Solutions are optimal in the abstract graph but not necessarily in the initial problem graph. Therefore, we perform a postprocessing phase for path smoothing to improve the solution quality. The main idea is to replace local suboptimal parts of the solution by straight lines. Starting from one end of the solution, for each cell, we check whether we can reach a subsequent cell in the path in a straight line. If this happens, then the linear path between the two cells replaces the initial suboptimal sequence between these cells [5]. This step could be done one frame after applying A*. If the entity begins to walk in the same frame as the proper A* or one frame later, it can hardly be recognized by the player.

3.2.3. Multilevel hierarchy

Additional levels in the hierarchy can reduce the search effort, especially for large mazes. In a multilevel graph, nodes and edges have labels showing their levels in the abstraction hierarchy. Pathfinding is performed using a combination of small searches at various abstraction levels. We build each new level on top of the existing structure. The clusters for level l are called l-clusters [5]. To search for a path between S and G, we search only at the highest abstraction level and will always find a solution, assuming that one exists. The result of this search is a sequence of nodes at the highest abstraction level. If desired, the abstract path can repeatedly be refined until the low-level solution is obtained.

3.2.4. Data structure representation

Looking at HPA*, we notice that the number of search operations in one pathfinding can be up-to $l + 1$ times: one search for the highest abstraction level and l searches for recursive path refinement. Even when caching and unit grouping are used, HPA* is still slow if the A* search operation is not efficient. To optimize A* search, we focus on improving the data structure representation.

Node and cell structure

The elements for pathfinding in this approach are nodes (for multilevel abstract graphs) and cells (for low-level graph). For simplicity, we can call these elements as cells. In A* search, the algorithm has the choice of connected cells from the current entity position. When it decides to go to a direction, it can choose again out of its connected cells and can calculate again. It goes on and on until one of the cells leads direct to the goal. Once the search reaches G, the algorithm has to trace back to S. Heuristic that could help us with probabilities, but an exact statement about which cells lead to the goal could not be made. That means all the cells have to be saved and to be recallable every time. Alternatively, we could search a path from G to S so that the path can be used immediately. Otherwise, the saved path has to be reversed. As the number of cells is large, it would be useful when our algorithm could process as much cells as possible to find even longer and complex ways. Here is some information a cell must contain.

(i) The position of the cell to calculate the distance to G, we may take the coordinate of its center point.

(ii) The heuristic to determine how probable it is to reach G from the current state of position.

(iii) A reference to the previous (parent) cell to trace back.

(iv) A unique ID: an individual number of identification for access every cell later on. It has to be approachable.

For example, with the low-level graph, every terrain consists of vertices which are numbered consecutively so that each vertex has its own unique number. Besides, most of the

engines have function to access the vertex directly based on its number. Therefore, the solution is to assign the unique number of the cell's center vertex to the cell's unique ID. There are alternative ways when we do not want to analyze the terrain in our game. However, we believe that pathfinding based on analyzing the terrain has better quality. Here is an example of defining cell:

$$\begin{aligned} \text{CELL[ID]} &= \text{cell\_center\_vertex\_number;} \\ \text{CELL[waycosts]} &= \text{PARENT\_CELL[waycosts]} + 1; \\ \text{CELL[cellcosts]} &= \text{CELL[waycosts]} \\ &\quad + \text{distance(current\_pos, goal\_pos);} \\ \text{CELL[parent]} &= \text{parent\_cell\_ID.} \end{aligned}$$

The information about the position of the cell can be found out through cell ID. Every time new cells get created, the waycosts increases by 1. The sum out of many heuristic values gets normally summarized as cellcosts. As an array represents a single cell, multidimensional array is used to represent the level grid. On the basis of the cellcosts, the algorithm has to go for a cell with the lowest cellcosts inside the array where the pathfinding continues. It would be very ineffective to let the algorithm search again in its saved cells for the best one since it already searched and saved the cells that lead to G. It would be more luxurious if the array with the saved cells is prearranged so that the presently cheapest cell is always at the first array entry. Among all the sort algorithms, the *heapsort* is the most efficient.

Heapsort

According to Williams [7], who invented Heapsort and the heap data structure, Heapsort is a combination of tree sort developed by Floyd [8] and tournament sort described, for instance, by Iverson [[9], Section 6.4] (see also [10]). A heap is a binary tree (a tree structure, whose knots have only two edges), whose roots/knots have a lesser (or greater, depending on heap attribute) value than their direct succession roots/knots. The heap attribute is determined by the heap order. When roots/knots have a lesser value than their successors, it is an increasing heap order (the deeper you go down the binary tree, the greater the value gets).

At a heap with increasing heap order like the example in Figure 15, the smallest value of this data structure always inside the root that is pretty practical because our array with the cell entries could sort the cellcosts that way—the presently cheapest cell (the cell with the least cell costs) would always be at the root. To represent the array as a heap structure, first, we put the first cell at CELL_LIST array on position CELL_LIST[1]; then, the successors of a cell in CELL_LIST[i] are saved at the positions CELL_LIST[2*i] and CELL_LIST[2*i + 1]. Reversely, the parent cell can be found by dividing the position of the current cell by 2: CELL_LIST[i/2]. In array shape, a heap would look like Figure 16.

We use the heap from the start as a data structure. The heap is not empty at the beginning; the heapsort sorts a new value directly after the entry. Also, changing and deleting

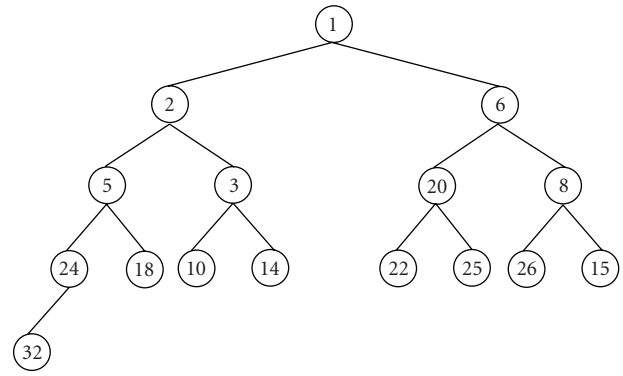


Figure 15: A heap with an increasing heap-order.

an entry (and the combined rearrangement) have to be managed by the heapsort. A heap that is used in such a kind of heapsort is called priority queue. The priority lays on the cellcosts that shall be possibly low. To add new cell or modify the value of a cell lesser, a procedure called *up-heap* [7] is used. The new cell is added as leafs at the end of the array, and the heapsort starts bottom-up. In case our defined heap order is overridden, the modified value of a cell is greater than the value of one of its child nodes, we have to use *down-heap* [7] procedure, sort top-down after the up-heap. We may optimize the sort by using other variants of heapsort such as weak heapsort [11] or ultimate heapsort [12].

3.2.5. Experimental results

In [5], experiments were performed on a set of 120 maps extracted from BioWare's game, BALDUR'S GATE, varying in size from 50 × 50 to 320 × 320. For each map, 100 searches were run using randomly generated S and G pairs where a valid path between the two locations existed. The experimental results show a great reduction of the search effort. Compared to a highly-optimized A*, HPA* is shown to be up to 10 times faster, while finding paths that are within 1% of optimal.

Figure 18 compares low-level A* to abstract search on hierarchies with the maximal level set to 1, 2, and 3. The left graph shows the number of expanded nodes and the right graph shows the time. For hierarchical search, the total effort is displayed, which includes inserting S and G into the graph, searching at the highest level and refining the path. The real effort can be smaller since the cost of inserting S or G can be amortized for many searches, and path refinement is not always necessary. The graphs show that, when complete processing is necessary, the first abstraction level is good enough for the map sizes that we used in this experiment. We assume that, for larger maps, the benefits of more levels would be more significant. The complexity reduction can become larger than the overhead for adding the level. More levels are also useful when path refinement is not necessary, and S or G can be used for several searches. Figure 19 shows how the total effort for hierarchical search is composed of the

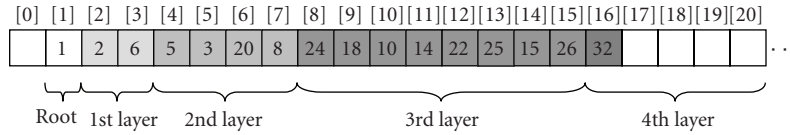


Figure 16: Representation of a heap in an array.

abstract effort, the effort for inserting S and G, and the effort for solution refinement. The cost for finding an abstract path is the sum of only the main cost and the cost for inserting S and G. When S or G is reused for many searches, only part of this cost counts for the abstract cost of a problem. Considering these, the figure shows that finding an abstract path becomes easier in hierarchies with more levels.

4. AI-BASED LEARNING

In Section 3, we discuss the approach to pathfinding used in MLA (Figure 2). In this section, we describe an AI-based learning design (Figure 3) to be used in all the agents. The purpose of AI-based learning is to capture and consolidate expert knowledge to achieve realistic game, evaluate the scenarios and strategies with greater accuracy. It will help the player experience increasing level of intelligence with every interaction in the game.

As RL is rule based, all the rules of the game will be extracted and stored in a *Rule Database*. During the game play, HLA would query the rules through *Rule API* from time to time. These rules will be used by computer's forces to play against the player. The detailed environment parameters and the result of action performed by agents are captured and logged in an *Exercise Database* which will be used for RL. In an offline situation where the game is not running, *Machine Learning* module analyzes the data from the *Exercise Database* based on RL functions and creates new rules or modifies existing rules for *Rule Database*. The modification of rules will increase AI gradually. It means that the level of difficulty rises up, and the player will find it harder to beat the computer [13]. Another function for the offline situation is the *Rule Editor* that has the capabilities to display, create, modify, and delete rules.

4.1. Rule API and rule database

Rule API is the interface for all operations. The most important functions are to attach *Rule Database* and to query the rules. When the game is loaded, each entity will be attached to its corresponding rule database through its agent.Subsequently, the entities can query for the rules in the rule database.The rules have to decide the actions to be carried out by the entities based on the information provided. Each query of the rule database will return one action. After the execution of that action, query the database for the next action will base on new information. As querying the database may become speed bottleneck, we may cache the entire rule database if the memory is large enough.

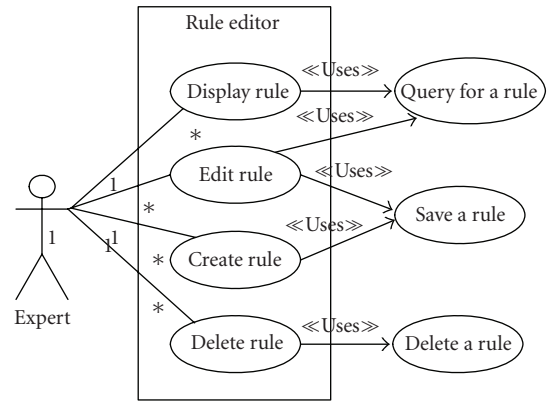


Figure 17: Use case for rule editor.

Otherwise, we only cache some of the frequently accessed rules.

In *Rule Database* , there are rules to define the *mission* of the forces which is the overall objective of HLA. This mission contains a set of *submissions* (SM) that is to be carried out by lower agents in order to accomplish the mission. For each command or overall mission, there is a set of SM that would be directly related to the mission stored in the database. The SM has to be assigned to forces to execute or complete the task. Information regarding the SM, for example, parameters, type of forces to be assigned, and priorities will also be provided in the database. Hence, there will be a rule database for HLA to assign the SM to forces. The assignment is under these conditions: after a main command (or overall mission) is given, a force has finished its assigned SM, new force is created, or a situation occurs, for example, enemy situation, operational situation, or obstacle situation. In the situation awareness, the mission, situation and its parameters are required by the rules. When a force encounters a situation, it will immediately react based on its rules of situation awareness. Hence, a rule database for MLA and LLA would also be necessary to respond accordingly. At the same time, the encountered situation and actions taken will also be reported to HLA which would then evaluate the situation and react appropriately. New SM can be reassigned to other forces whenever necessary. If the situation is not resolvable by the rules, user intervention may be requested.

4.2. AI agent

Every entity that is said to have AI will have an *AI agent* assigned to it. At LLA, AI called *unit agent* is used to control

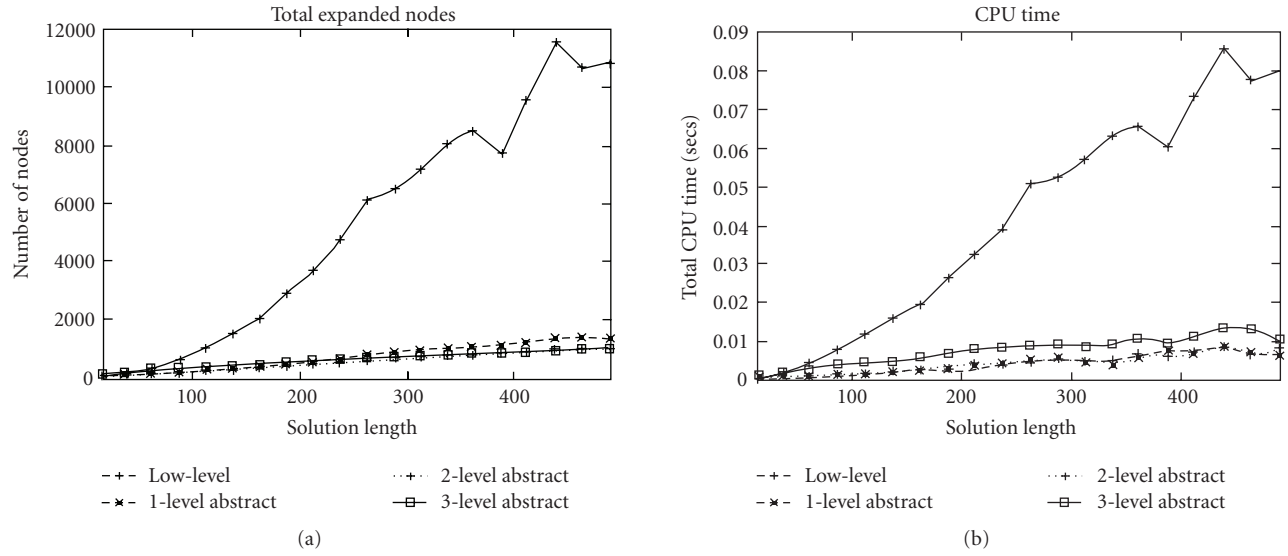


Figure 18: Low-level A* versus hierarchical pathfinding.

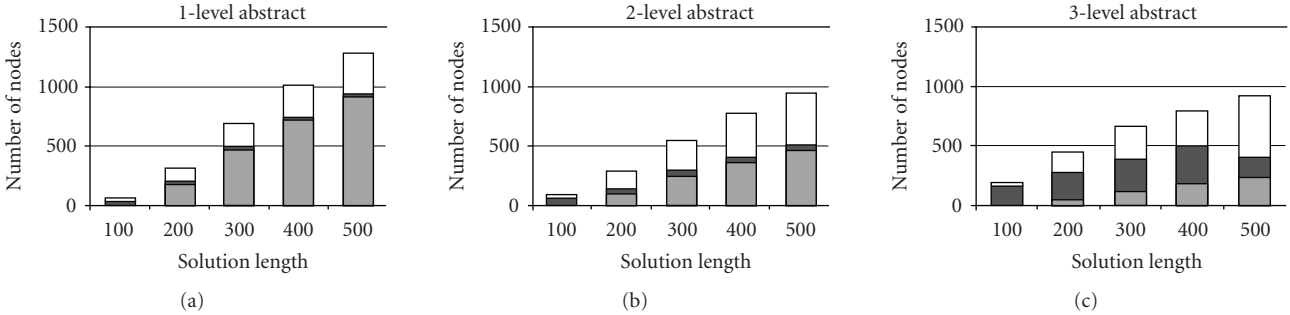


Figure 19: The effort for hierarchical search in hierarchies with one abstract level, two abstract levels, and three abstract levels. We show in what proportion the main effort, the SG effort, and the refinement effort contribute to the total effort. The gray part at the bottom of a data bar represents the main effort. The dark part in the middle is the SG effort. The white part at the top is the refinement effort.

detailed behaviors of different units. Unit agent is attached to every unit entity and consisted of various state machines to handle the detail movement and strategic reactions when it carries on the task given. Detail movement of an entity is determined by the game mechanics such as stand, guard, run, shoot, and throw grenade. Strategic reactions consist of individual reaction and group reaction.

4.2.1. Individual reaction

The unit agent will consider its survival probability as well as the present of enemy force in its line of sight to act according to the situation.For example, consider the case when a unit is at state *stand*, it detects enemy within its range of fire. If no task is given by higher agent, the unit agent has choices to 50% switch to state *shoot*, or to 30% switch to state *retreat*, or to 20% remain in state *stand*. The probability parameters are specified in the rule database and are loaded into the game at initialization process. We notice that game difficulty level could be adjusted simply based on some factors that affect the "skill" of the unit agent. For example, the reaction time, update cycle speed, health level, fire power of the enemy forces could be increased to add in challenges. The opponent could also have a "cheat" factor, that is, it will be given more units than the player.

4.2.2. Group reaction

Agents will also be attached to capture the hierarchy, that is, battalion, company, and the group behaviors. These agents will communicate with unit agents to get the status of different units. This status will help the hierarchical agent to make a better decision. For example, group formation in movement is useful to ensure that all the units keep their original formations upon reaching their targets. To achieve group formation, we use a simple approach: calculate the center position of all the selected units (a point that is roughly the middle of where they currently are). From that point, we get the offset for each unit, for example, if the center point is at [5,1] and one unit is at [6,1] then the offset would be [1,0]. The offset is, then, added to the destination point and that would be the point to move the selected

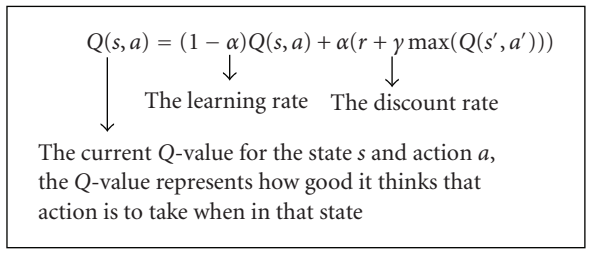


Figure 20

unit to. This would ensure all the units keep their original formations upon reaching their targets.

Another example is *coordinated behavior* in the enemy situation. In case our units surround the enemies, we want them to shoot the enemies without shooting at each other. Some of the games make it simple by letting the bullet go through ally to reach the enemies. We also can implement a small procedure to avoid friend's line of fire. In the enemy engagement, if a unit has line of sight to the target, it can shoot immediately. Otherwise, if obstructing object is an ally, request him to move away. If the ally is busy, or obstruction is not an ally, the unit moves itself to another place until it has the line of sight to the target. Another possible solution would be flocking which lets units repulse from each other and arrives at different offsets from the destination. However, we believe that flocking is overkill for RTS game, unless we really want to mimic the behavior of flocks.

4.3. Rule editor

The main responsibility of *Rule editor* is to edit the rule database. It has functions to add, delete, and modify any rule. It follows the model-view-controller design pattern. The Editor facilitates to change the state of the database on receiving instructions. In Figure 17, *Display Rule* allows the experts to view the rules displayed sequentially within a rule database. The expert has the ability to skip the current database and view the rules from another one. *Edit Rule* is to edit an existing rule. *Create Rule* allows the expert to create new rule from scratch. The expert, then, key-in or select the desired values for the parameters as well as the actions or output the rule would return. The completed rule will be stored in respective rule database. While the existing rules are displayed, the expert is given the option to delete the rule using *Delete Rule*.

4.4. Machine learning

This module is used to learn from environments, scenarios, and unsuccessful attempts. Based on the information obtained, it would try to extract new rules. The module checks with the rule database to ensure that the learnt rules are not present in the database. Newly learnt rules would be saved to the database [13].

RL function, Q-Learning [14], uses "rewards" and "punishments" so that an AI agent will adapt and learn appropriate behaviors under some conditions. In the experience tuple (s, a, r, s'), s is the start state, a is the action taken, r is the reinforcement value, and s' is the resulting state. The exploration strategy is to select the action with the highest Q-value from the current state.

There are two types of learning: supervised learning and unsupervised learning [15]. *Supervised Learning (SL)* is when machine learns from the user through user's input or adjustment of parameters. SL occurs when the rules fail to decide on an appropriate reaction to a situation and request for user's intervention or when the user decides to intervene. This intervention and its result will be logged in the *Exercise Database* for the offline learning. During the offline learning process, the effectiveness of user intervention is analyzed, and a new rule is generated.

Unsupervised Learning (UL) , in contrast, is to learn new rules without the knowledge or inputs from the user. The learning would be based on the existing set of rules to either generate new rules or enhance the old one. Some rules are specific to be fired by certain situations; some are more generic to be fired by a larger number of situations. The situations that would fire the rules could interest or subset with another one. This may result in several possible rules to be fired for one situation. Hence, these possible rules for a situation need to be prioritized to obtain the most efficient outcome. For example, the assignment of SM to forces can be conducted in several ways or sequences, and UL is to learn the best way to assign the SM. Each possible assignment of forces is valued with a priority or probabilities. Usually the possible assignment with the highest value is selected. If a sequence of assignment fails in a mission, the probabilities of this sequence will be decreased accordingly to reflect the failure. On the other hand, the probability would increase for a successful mission. Similar concept is also applied to the rules that respond to situations. Rules will be rewarded or punished based on the successful or failure executions of the reactions.

5. CONCLUSIONS

This research work is from the development of the basic game with simple AI modules, to the research of the higher-level concepts—advanced AI-based learning algorithms. Using the game demo as an effective tool, we implement various game AI techniques such as finite state machine, group behaviors, and pathfinding algorithms. We, then, work on finding the optimal combination of efficient techniques that are easy to implement and generic enough to be applicable in many games with little implementation changes. Based on this combination, we design the architecture for RL and propose the framework for future developments.

Our approach can have any number of hierarchical levels, making it scalable for large problem spaces. When the problem map is large, a larger number of levels can be the answer for reducing the search effort, for the price of more storage and preprocessing time. We use no application specific knowledge and apply the technique independently of the map properties. We handle variable cost terrains and various topology types such as forests, open areas with

obstacles of any shape, or building interiors without any implementation changes.

This research work has exposed us to new technologies and to current trends in computer game industry. We have explored some of game AI techniques and evaluated their pros and cons as part of the objectives. These technologies have shown to possess great potential in penetrating into the market, and there is plenty of room for improvement.

In the future, we will continue evaluating the proposed RL architecture to prove its effectiveness. We will also explore on some advanced techniques such as fuzzy logic, Bayesian networks, and neural networks, and will modify them to use in strategic game domain. Using these techniques, we will focus on tactical AI, particularly focusing on pathfinding, tactic analysis, and tactical representation. In addition, group dynamics and coordinated behavior are also very interesting to spend time on. At the same time, the underlying cognitive architecture needs to be expanded to make the games even more realistic.

REFERENCES

[1] K. D. Forbus, J. V. Mahoney, and K. Dill, "How qualitative spatial reasoning can improve strategy game AIs," *IEEE Intelligent Systems*, vol. 17, no. 4, pp. 25–30, 2002.

[2] E. Bethke, *Game Development and Production*, Wordware, Plano, Tex, USA, 2003.

[3] K. Forbus, "Qualitative reasoning," in *CRC Handbook of Computer Science and Engineering*, pp. 715–733, CRC Press, Boca Raton, Fla, USA, 1996.

[4] A. G. Cohn, "Qualitative spatial representation and reasoning techniques," in *Proceedings of the 21st Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence (KI '97)*, vol. 1303 of *Lecture Notes in Computer Science*, pp. 1–30, Springer, Freiburg, Germany, September 1997.

[5] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding," *Journal of Game Development*, vol. 1, no. 1, pp. 7–28, 2004.

[6] S. Rabin, "A* speed optimizations," in *Game Programming Gems*, M. DeLoura, Ed., pp. 272–287, Charles River Media, Rockland, Mass, USA, 2000.

[7] J. W. J. Williams, "Algorithm 232: heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, 1964.

[8] R. W. Floyd, "Algorithm 113: treesort," *Communications of the ACM*, vol. 5, no. 8, p. 434, 1962.

[9] k. E. Iverson, "A programming Language," John Wiley and Sons, New York, NY, USA, 1962.

[10] E. H. Friend, "Sorting on electronic computer systems," *Journal of the ACM*, vol. 3, no. 3, pp. 134–168, 1956.

[11] R. D. Dutton, "Weak-heap sort," *BIT Numerical Mathematics*, vol. 33, no. 3, pp. 372–381, 1993.

[12] J. Katajainen, "The ultimate heapsort," *Australian Computer Science Communications*, vol. 20, no. 3, pp. 87–95, 1995.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Mass, USA, 1998.

[14] I. Millington, *Artificial Intelligence for Games*, Morgan Kaufmann, San Mateo, Calif, USA, 2006.

[15] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, Prentice Hall, Upper Saddle River, NJ, USA, 1994.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 432365, 18 pages
doi:10.1155/2008/432365

Research Article

A Hybrid Fuzzy ANN System for Agent Adaptation in a First Person Shooter

Abdennour El Rhalibi and Madjid Merabti

School of Computing and Mathematical Sciences, Liverpool John Moores University, James Parsons Building, Byrom Street, L3 3AF, Liverpool, UK

Correspondence should be addressed to Abdennour El Rhalibi, a.elrhalibi@ljmu.ac.uk

Received 31 July 2007; Accepted 1 November 2007

Recommended by Kok Wai Wong

The aim of developing an agent, that is able to adapt its actions in response to their effectiveness within the game, provides the basis for the research presented in this paper. It investigates how adaptation can be applied through the use of a hybrid of AI technologies. The system developed uses the predefined behaviours of a finite-state machine and fuzzy logic system combined with the learning capabilities of a neural computing. The system adapts specific behaviours that are central to the performance of the bot (a computer-controlled player that simulates a human opponent) in the game with the paper's main focus being on that of the weapon selection behaviour, selecting the best weapon for the current situation. As a development platform, the project makes use of the Quake 3 Arena engine, modifying the original bot AI to integrate the adaptive technologies.

1. INTRODUCTION

With graphics at an almost photorealistic level and complex physics systems becoming commonplace, AI [1] is becoming more important in providing realism in games.In the past, game AI has used techniques that are suited to the restricted computational power available to it, but which still produced believable, but limited, nonplayer characters (NPCs) artificial intelligence (AI) technologies such as finite state machines (FSM) and rule-based systems (RBS). These techniques were also used due to their relative simplicity which did not require much development time to implement and were easy to debug, especially as the programmers generally did not specialise in AI.

With the increase in computational power available for AI, more complex techniques can be incorporated into games creating more complex behaviours for NPCs. The increasing importance of AI in games has meant that specialised AI programmers are becoming part of development teams bringing techniques from academia [2–5]. One of the areas of AI which has gathered interest is that of using machine learning techniques to create more complex NPC behaviours.

Most players develop styles of play that take advantage of certain weaknesses inherent in the NPC AI that become apparent as they become more proficient at the game. Once discovered, these deficiencies in the preprogrammed AI mean the competitive edge is lost making the player lose interest in the now all too easy game. If the NPC developed new tactics, adapting to the players style, uncovered their hiding places, or even discovered tactics that exploited weaknesses in the players' play, then this would add immeasurably to the enjoyment and prolong the life of the game [6–8]. The game should be tailored to provide a variety of challenges, and increasing the level of difficulty to deal with NPCs. This should of course be adjusted to the need of the player and be provided as a way to increase difficulty level in the gameplay without introducing unbeatable NPC which could lead to player's frustration. We are aware that it is a difficult issue to adjust the balance gameplay between performance and playability, and it will be the role of the game designer to deal with it. We are just interested in this paper in increasing NPCs' performance and adaptation.

This paper describes a method of implementing a first-person shooter (FPS) bot which uses machine learning [9] to adapt its behaviour to the playing style of its opponent. It

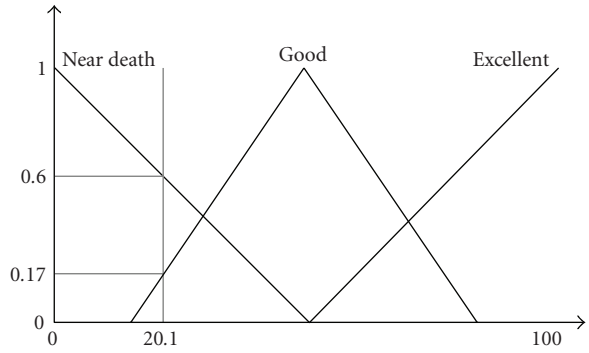


Figure 1: Sets defining the linguistic variable "health."

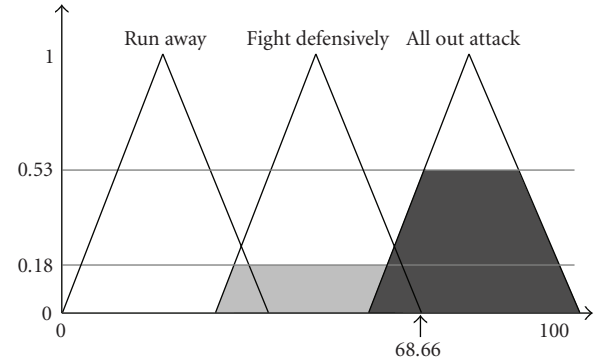


Figure 2: Sets defining the output variable "aggressiveness."

uses a combination of small, focussed, artificial neural techniques and predefined behaviours that allow the bot to exhibit changes in those behaviours to compensate for different player styles. For the purposes of this paper, only a single behaviour is focused on, that of weapon selection, although, in order to significantly adapt the play of the bot, multiple behaviours would use the system during a match.

2. FOUNDATION TECHNOLOGIES

2.1. Fuzzy logic

Whereas traditional logic describes concepts in terms of "true" or "false," fuzzy logic provides a way of describing values by the degree with which they correspond to a certain category within the concept, called DOM (the degree of membership) in a *set*. *Linguistic variables* are collections of sets that represent real concepts, for example the variable *health* could be made up of the sets near death, good, and excellent, as shown in Figure 1 [3, 10, 11].

Fuzzy logic provides a way of combining more than one variable to give a single output value, making decisions based on multiple criteria. For example, the aggression of a game character based on its health and the distance to the enemy.

Using fuzzy logic to derive decisions based on the input values for a number of variables requires that a sequence of steps to be carried out.

(1) Selection of sets that comprise the linguistic variables for the inputs and output. As with the input variables, the output variable consists of a number of sets defined by a range of values (see Figure 2). The difference is in the way they are used to calculate the final out value (see step (6), *Defuzzification*).
(2) Creation of fuzzy rules corresponding to the different combinations of inputs. The rules determine the output set for the different combinations of inputs. Using the previous example, if health is "good" and distance is "close," the rule could be "fight defensively."
(3) *Fuzzification* of the crisp inputs into fuzzy values giving the DOM for the inputs sets. Figure 1 shows the fuzzification of the crisp value 20.1 resulting in the DOM for each set of near death = 0.6, good = 0.17 and excellent = 0.0.
(4) Use *inference* to evaluate which rules are active based on the DOM of the input sets that make up that rule. Each combination of sets (for each input variable) is compared with the rulebase to determine which output sets are active. The DOM of the output set is determined, in this case, using the lowest DOM of the inputs (there are a number of methods for calculating the DOM). This results in a number of possible DOMs for each set of the ouput variable.
(5) Combine the multiple DOMs for each rule into the output sets using *composition*. This results in a single DOM for each of the output sets, as shown in Figure 2.
(6) *Defuzzification* of the output sets to give a single crisp value. This is done by calculating the centre of the area under the graph defined by the DOM in each set. Figure 2 shows an example for the output variable "aggressiveness" with doms of 0.18 for "fight defensively" and 0.53 for "all out attack." There are a number of methods, of varying complexity and accuracy, for determining the output value. One of the least expensive, in terms of computation, is the *mean of maximum* method, the equation for which is shown in (calculation of mean of maximum)

$$\frac{(\mathrm{RA}_{\max} * \mathrm{RA}_{\mathrm{dom}} + \mathrm{FD}_{\max} * \mathrm{FD}_{\mathrm{dom}} + \mathrm{AA}_{\max} * \mathrm{AA}_{\mathrm{dom}})}{(\mathrm{RA}_{\mathrm{dom}} + \mathrm{FD}_{\mathrm{dom}} + \mathrm{AA}_{\mathrm{dom}})}, \tag{1}$$

where

(i) $\mathrm{RA}_{\max}$ is the crisp value for the centre of "run away" set (where fuzzy value = 1);
(ii) $\mathrm{RA}_{\mathrm{dom}}$ is the fuzzy value for DOM for run away set;
(iii) $\mathrm{FD}_{\max}$ and $\mathrm{FD}_{\mathrm{dom}}$ are as above for "fight defensively" set;
(iv) $\mathrm{AA}_{\max}$ and $\mathrm{AA}_{\mathrm{dom}}$ are as above for "all-out attack" set.

2.1.1. Combs method

In traditional fuzzy logic, a rule needs to be defined for every combination of set for all the input variables. This can result in combinatorial explosion as the number of rules required grows exponentially according to the number of fuzzy sets for

each linguistic variable, that is, 2 variables each with 5 sets = 5^2 = 25 rules and 5 variables with 5 sets = 5^5 = 3,125 rules. This can make large systems slow, confusing, and difficult to maintain which, particularly in games, can make fuzzy logic impractical.

The main difference between Combs method [10] and the traditional method is in the way the rule set is defined. It builds rules based on each individual set's relationship to the output, considering one variable at a time, rather than creating rules for every combination of set for all the variables. This reduces the exponential growth of the number of rules into a linear growth, so that a system with 10 variables and 5 sets per variable would have 50 rules as opposed to 9 765 625 with the traditional system.

2.2. Artificial neural networks

There are many forms of artificial neural nets (ANN) of varying complexity which attempt to mimic the biological operation of the brain artificially by modelling the inter-connected cells that enable the brain to process information. The simplest form of ANN, the one used here, is the perceptron which is modelled as a single neuron with a set of weighted inputs mapping to a single output [7, 12–14].

The inputs (X_1 to X_n) to the perceptron can vary in number and value (binary or real numbers) depending on the application. Each input is multiplied by its corresponding weight (W_1 to W_n) and the weighted inputs are then added together, along with the bias, giving the output value. The bias represents a constant offset and can be treated as another input with a constant value of 1. By adjusting its weights, the perceptron can be trained to recognise specific combinations of inputs and generalise for similar inputs.

2.2.1. Training the perceptron

Initially, the perceptrons use a default value for all of their weights. This, in effect, means that the perceptrons will not have any influence over the effectiveness rating for the weapons, only the characteristics and fuzzy logic will affect the value. Once adaptation has begun, occurring every time there is feedback, the following training procedure is performed.

The training of perceptrons described here uses an incremental approach, computing the adjustments to the weights by way of the steepest descent technique [7]. The *delta rule* algorithm calculates the change required Δw_i for each weight w_i by taking the difference between the actual y and the desired t output and multiplying it by the input value x_i for that weight and by a, typically small, learning rate η; see (2), computation of required adjustment for each weight:

$$\Delta w_i = \eta(t - y)x_i. \tag{2}$$

The new weight for each input can then be found using the *steepest descent technique*, as shown in (3), computation of adjusted weight value, changing the weights as a result of feedback:

$$w_i \longleftarrow w_{i+}\Delta w_i. \tag{3}$$

The incremental nature of the algorithm means that it can be performed as the game is being played using feedback from actions performed.

2.3. Quake 3 arena

In order to implement the adaptable AI, a suitable environment was required that provided all the features of a FPS so that the capabilities of the bot can be tested. The Quake 3 Arena (Q3A) game engine [8, 15] provided the framework for the development of the bot AI. The new AI was integrated with the original AI, reusing many of its features. For more information regarding the Q3A engine, specifically in relation to the interface between the AI and the game engine, [8] provides the most comprehensive documentation.

Using the original bot AI provided the opportunity to be able to define the characteristics of the bot using text files that determine the style of play of the bots within the game. This proved helpful in the evaluation of the new AI, which was carried out in matches against the "standard" Q3A bots. By defining specific characteristics, situations could be set up that required the bot to adapt its behaviour.

3. SYSTEM DESIGN

A number of features are required of the adaptable AI system in order to achieve the aim of a bot that is able to adapt to the play of an adversary:

(i) to be able to play competitively from the first game (out of the box);
(ii) to adapt its behaviour as the game is being played (online);
(iii) to be computationally inexpensive.

The system makes use of the indirect adaptation technique, using a conventional AI layer to control the bot, with the adaptation AI modifying the behaviours of the bot in response to feedback according to its actions. This enables the bot to be competitive immediately by giving it a priori knowledge, as recommended by [1, 3, 5].

The adaptation system incorporates a number of components that combine to rate the effectiveness of a choice within a behaviour and adapt the value to reflect how well the chosen action performs in the game. Figure 4 shows how the separate elements are linked together to calculate the rating and allow adaptation to occur.

The system utilises a hybrid of two AI technologies: fuzzy logic and perceptrons. The fuzzy logic acts as the prior knowledge enabling the bot to perform in the game at a competitive level. The perceptron is used to facilitate the adaptation, acting as a form of memory enabling the bot to "remember" the effectiveness of actions in certain situations, altering its weights based on the feedback it receives from game. By using perceptrons, rather than more complex multilayer networks, the computational requirements are kept as low as possible whilst retaining the basic features of a neural net.

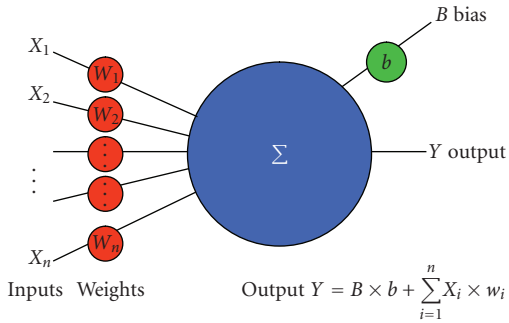


Figure 3: Architecture of a perceptron.

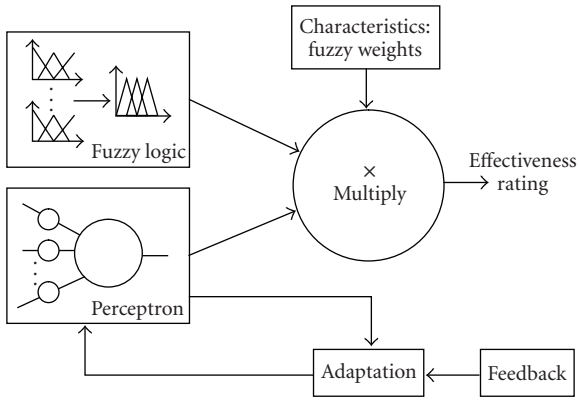


Figure 4: Adaptation system overview.

The system is composed of two main mechanisms:

(i) the effectiveness rating mechanism: used to determine how effective a certain choice is according to the input values;

(ii) the adaptation mechanism: used to change the effectiveness rating according to feedback from the game on how effective it was.

The effectiveness of a choice is predicted using a combination of the characteristics of the bot, defined in the characteristic files, a fuzzy logic component and a perceptron component. This system is used for each of the choices within a behaviour. The effectiveness is calculated by multiplying the outputs from the fuzzy logic component and the perceptron together with the characteristic for the choice.

The adaptation mechanism uses feedback from the game to determine how successful the choice was compared to the perceptron's predicted effectiveness of the choice. The feedback and output of the perceptron are then used to train the perceptron, increasing or decreasing the weight values according to the delta rule training algorithm discussed in Section 2.1.1. Adjusting the weights of the perceptron changes its output impacting on the effectiveness rating for the action, thus making it more or less likely to be used.

3.1. Adaptation of weapon selection behaviour

Modern FPS games, such as Quake 3 Arena, make use of complex 3D environments for their game worlds which, in turn, mean that the NPCs that inhabit them must have complex AI to interact with them, and the player, convincingly. Bots must be able to exhibit a number of behaviours, specialising in particular actions or strategies that contribute to the overall aim of winning the game. Due to the nature of the game, the aim being to kill the opponent more times than they kill you, the behaviours that would benefit most from adaptation are those that relate to combat with opponents, either directly or indirectly. One such behaviour is that of selecting the most effective weapon for the current situation. The rest of the paper will focus on this behaviour to demonstrate how the system can be applied.

The aim of adapting the selection of weapons is to enable the bot to change its weapon preferences depending on its success in particular situations. By changing the "effectiveness" or "fitness" of each weapon, by way of changing the perceptron weights according to the input values, different play styles can be adapted to.

The selection of information used as inputs for the system components is vital to their efficiency at performing actions in the game. The following sections detail the inputs for the fuzzy logic and perceptron components.

3.1.1. Fuzzy logic for weapon selection

Each of the weapons have a set of data defined for the variables (inputs) that represent the range of values that are significant to that weapon. The variables used for the fuzzy logic component are the following.

(i) Distance to the enemy. Each of the weapons available is better or worse at different distances. For example, the Lightning Gun has a maximum range of 768 and the Rocket Launcher risks splash damage when used at close distance. The distance needs to be broken down into fuzzy sets defining the effectiveness of each weapon for the distance range represented by that set.

(ii) Ammunition amount for each weapon. Each of the weapons have different firing rates. For example, the Machine Gun fires a shot every 1/10th of a second whilst the Railgun can only fire a shot every 1.5 seconds.Running out of ammunition in a fight means changing to another weapon, which takes time, reducing the damage that can be inflicted on the enemy. The ammunition level needs to be represented as a number of fuzzy sets spanning the maximum amount of ammunition (200). Each weapon requires a unique collection of set data defining the relative values of ammunition depending on their rates of fire—10 ammunition for the Railgun is different to the same amount for the Machine Gun.

3.1.2. Perceptron for weapon selection

Each opponent and game map have their own set of perceptrons as, for instance, different weapons can be more or less

effective depending on the map being played. Each weapon is represented by a perceptron, each having a unique set of weight values for that weapon. The inputs to the perceptron are the same for each of the weapons, although weapon specific inputs, that is, the amount of ammunition, will result in certain inputs having slightly different values. Some of the variables investigated are the folllowing.

(i) Distance to the enemy. By adapting the distance at which the weapon should be used, the weapons will increase/decrease the range at which they are used. An example of a use for this is if the enemy is very aggressive and continues attacking when low on health. Normally the rocket launcher may not be used at close range due to the danger of splash damage, selecting a less damaging, and less successful, weapon instead. The system could adapt the lower range of the Rocket Launcher so that it is selected over the less useful weapon, incurring damage to the bot but also killing the opponent with one shot.

(ii) Ammunition. The amount of ammunition for each weapon can be adapted to make use of weapons that the opponent is more susceptible to be damaged by. Used by the fuzzy logic component, it has a large influence on the selection of weapons and by adjusting the ranges the bot will be more likely to stick with a successful weapon even though the ammunition is running low in the hope of killing them before the weapon needs to be switched.

(iii) Visibility of enemy. It would be useful to adapt the weapon selection based on the visibility of the enemy so that areas that contain obstacles, creating cover for the enemy to hide behind, can influence the selection to favour weapons that have splash damage enabling the weapon to inflict damage around corners.

(iv) Height difference. Like the visibility of the enemy, the height difference between the bot and its opponent could be used to influence the use of weapons that have splash damage. If the opponent is below the bot, it can aim at the floor near to the enemy, hitting with radial damage. If the opponent is higher, making it difficult to hit them with splash damage, then Grenades can be launched onto the higher area or more precise weapons can be used. Adapting the relative strengths of weapons when there is a height difference will select the most effective weapons in those situations.

3.2. Feedback for perceptron training

The feedback that is used to train the perceptrons for the weapon selection behaviour is focused on the criteria of causing as much damage as possible whilst avoiding inflicting damage to oneself. This means that it must account for a combination of health lost by the enemy and by the bot itself as a result of its own attack (not damage sustained from enemy attack). A timed aspect is required to allow for the different characteristics of each of the weapons (firing rate and damage per shot) and enable the performance of the weapons to be compared. To reward weapons that have the capability of "finishing off" enemies (e.g., Railguns are very good at one-shot kills) a bonus is also required when the opponent is killed by the current weapon. This increases the overall feedback value thus increasing the weight values when training.

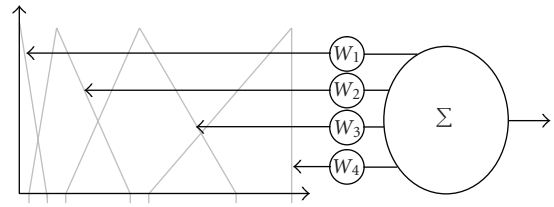


Figure 5: Categorisation of perceptron inputs.

3.3. Categorisation of perceptron inputs

Due to the linear nature of perceptrons (they are unable to handle nonlinear problems) difficulties arise with inputs that can be effective at high and/or low values. One problem is that higher input values will always output higher ratings and so if the lower input values are better (i.e., correspond to a more effective weapon use), these inputs values will not be able to characterize this effectiveness. Another problem is if the weapon is more effective with an input value that is in the middle range, such as the grenade launcher that can cause splash damage close-up but has a limited range. This is compounded by the training mechanism that changes the weight of the input depending on the input value. This means that high values will always be penalised more than low values.

To allow adaptation to occur independently for different levels of the same input, its range of values needs to be categorised into ranges. The fuzzy logic component can be utilised to achieve this. It is able to take a single value and assign a DOM for each of the categories by fuzzifying the input value. Each of the categories represents an input into the perceptron, splitting the single input value into the number of sets that represents that input, as shown in Figure 5. The advantage of this approach is that it will categorise the input into continuous values for each set, rather than the imprecise method of just determining whether the value is in a category or not. It also uses functionality that is already within the system so no new component needs to be developed.

One of the main advantages with using fuzzy logic to categorise the input value is that the fuzzy values will represent the DOM for the set. This means that the low category can have a high input value and the high a low value—0 (100% membership of the low category) could input a 1. When training the perceptron, this will be useful in correctly rewarding or punishing the value range responsible for the action selected. Another advantage is that the maximum membership of a set is 100%, in effect normalising the input values for each set to a value between 0 and 1. Although the input value can be in multiple sets, the combined fuzzy values will approximate 1 (fuzzy values need not add up to 1 but are usually near to this value, depending on the set data).

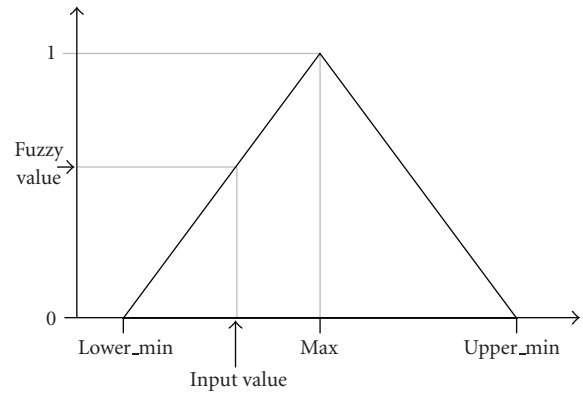


Figure 6: Data definitions for a fuzzy set.

4. DISCUSSION ON IMPLEMENTATION

4.1. Fuzzy logic component design

The fuzzy logic component is comprised of three parts. The first represents the fuzzification process, turning the crisp values of the inputs into fuzzy values of the degree of membership in the sets for that input, along with the rule associated with those sets. The second part, representing the composition process, calculates the degree of membership of the output sets based on the rules and fuzzy values calculated in step 1. The last part, the defuzzification process, determines the output value for the component.

It was decided during the design of the fuzzy logic component that each of the input variables should have four sets. In keeping with Combs method, the output sets should have the same number of sets as the input variables and so they also have four sets. Using four sets to define a variable provides a balance between sufficient detail to describe the inputs properly without making the component overly complicated.

4.1.1. Fuzzification process

The fuzzification of a crisp value into a fuzzy value or values is achieved using the data that defines the membership function for each set in the input variable. The data for the sets is defined in an array that is loaded during initialisation of the game. Figure 6 shows a typical representation of a fuzzy set with the data that represents that set labelled on the x-axis. The two "min" values represent the upper and lower limits of the set at the point where the degree of membership in that set equals 0. In between these two points the degree of membership will be greater than 0 with the maximum fuzzy value, 1, marked by "max."

Using this data, any input value that is within the "min" range can have a fuzzy value calculated for it using linear interpolation. The point at which the input value crosses the line defining the edge of the set (joining lower_min and max) can be determined by finding the difference between the input value and the max or min (depending on which is highest) and dividing it by the difference between the max and min values that the input value bisects. This results in two equations depending on whether the input value is between the lower_min and the max, or the upper_min and the max. Equation (5) presents calculation of the fuzzy value when max < input < upper_min,

$$\text{fuzzy value, fv}_l = \frac{(\text{input} - \text{lower\_min})}{(\text{max} - \text{lower\_min})}. \tag{4}$$

Equation (1) presents calculation of the fuzzy value when max < input < upper_min,

$$\text{fuzzy value, fv}_u = \frac{(\text{upper\_min} - \text{input})}{(\text{upper\_min} - \text{max})}. \tag{5}$$

Along with the fuzzy value, the rule associated with the set is also required so that the fuzzy value can be applied to the correct output set during composition. The rule is represented by the array location of the output set, that is, output [0] is bad, output [1] is average, [2] is good, and [3] is excellent. The fuzzification of the input value will result in 2 or 4 values being returned; the input usually has a degree of membership in 2 sets, 1 set only if the degree of membership is very high (>90%), so the fuzzy value and rules for both sets must be returned.

Pseudocode for fuzzification process

Algorithm 1 shows how the fuzzification process is calculated,calculating the fuzzy value if within a set, 1 if equal to the max value, and 0 if outside (also setting the rule to −1 to mark it as unused).

The fuzzification function is designed so that, as well as being used in the fuzzy logic component, it can also be used for single inputs when categorising input values for use with the perceptron.

4.1.2. Composition process

The composition of the fuzzy values and their associated rules into the degree of membership for each of the output sets is done by taking the MAX fuzzy value associated with each of the output sets calculated for all the inputs. This returns an array of values for each of the output sets that can be used to determine the output value in the next part, defuzzification.

Pseudocode for composition process

The process of composition is quite straightforward, simply putting fuzzy values into an array representing the output sets if the value is greater than the one currently in there. Algorithm 2 shows how this process can be accomplished.

(i) Fuzzy_values[] is an array containing the fuzzy values and rules calculated in the fuzzification process where [0] to [3] is the fuzzy values and rules for an input.
(ii) Output_array[] is an array that contains the MAX values for each of the output sets where [0] is set 1, [1] is set 2, and so forth.

```
for each of the inputs
  for each of the sets
    if set_lower_min value < input_value < set_max_value
      fuzzy_value = (1 /
                  (set_max_value − set_lower_min_value))*
                          (input_value − set_lower_min_value)
      output_rule = set_rule
  end of if
  else if set_max_value < input_value < set_upper_min_value
        fuzzy_value = (1 / (set_upper_min_value −
          set_max_value))*(set_upper_min_value − input_value)
        output_rule = set_rule
  end of if
      else if input_value = set_max_value
            fuzzy_value = 1
            output_rule = set_rule
      end of if
            else fuzzy_value = 0
                 output_rule = −1
            end of else
  end of for
end of for
```

Algorithm 1: Pseudocode for fuzzification process.

```
for number of sets (i)
    if fuzzy_values[i + 1] > −1
      if output_array [fuzzy_values[i + 1]] <
                        fuzzy_values[i]
       output_array [fuzzy_values[i + 1]] = fuzzy_values[i]
      end of if
      increment i by 2
end of for
```

Algorithm 2: Pseudocode for composition process.

```
for number of output sets (i)
    mean of max top + = output_data[i]* output_array[i]
    mean of max bottom + = output_array[i]
end of for
mean of max = mean of max top/mean of max bottom
```

Algorithm 3: Pseudocode for defuzzification process.

This process will be done for each of the inputs in turn; finally getting the MAX values for each of the output sets after all the inputs have been processed.

4.1.3. Defuzzification process

The defuzzification process takes the array generated by composition and returns the final crisp value that is used to determine the rating of the action. It uses the mean of maximum method of defuzzification to calculate the single output value, based on the max values of each of the output sets and the fuzzy values for that set.

Pseudocode for defuzzification process

The defuzzification process uses the output array (Output_array[]) and the stored data for the output sets (Output_data[]) to calculate the output value.

4.2. Perceptron component design

The perceptron component does not require a separate function to calculate its output value. A perceptron is a combination of a multiplication for each of the inputs and its associated weight followed by a sum of all the multiplications. The perceptron calculation is incorporated into the functions for each behaviour, as specific information for the inputs is required in each case.

4.2.1. Pseudocode for perceptron component

The code shown in Algorithm 4 shows the design of the perceptron component that is incorporated into each behaviour. A single function is not used for simplicity, as the requirements of each behaviour regarding the number of inputs to the perceptron and the information to get differ enough to warrant separate functions. Each of the functions follows the same design, just using different information.

```
get input values specific to behaviour
for each behaviour action (a)
        for each input to perceptron (i)
            output[a] + = input_value[i]* weight[i]
        end of for
end of for
```

Algorithm 4: Pseudocode for perceptron component.

```
SelectBestWeapon() function
        fzEval = EvalFuzzyWeapons()
        pEval = EvalPerceptWeapons()
        fzVal = GetWeaponFuzzyVals()
        for each weapon
            eval = FzEval* PEval* FzVal
            if weapon is current weapon
               eval *= 1.1
            end of if
            if eval is highest value
                best weapon = eval weapon
               end of if
        end of for
        return eval weapon
end of function
```

Algorithm 5: Pseudocode for weapon selection function.

4.3. Weapon selection behaviour design

The weapon selection behaviour is handled by a single function from which the fuzzy logic and perceptron evaluations are called and all the relevant input data extracted. This function replaces the original Q3A function *trap_BotChooseBestFightWeapon()* for the adaptable bot in the *BotChooseWeapon()* function. Algorithm 5 shows the structure of the function and how the effectiveness for each weapon is determined.

The variables fzEval, pEval, and fzVal are all arrays that are filled with the data for all the weapons stored in the same array locations in each, that is, array location [0] contains all gauntlet data, [1] machine gun data, and so forth. The fzEval array holds the fuzzy logic evaluations for each weapon, pEval the perceptron evaluations, and fzVal the fuzzy weapon weights defined in the characteristic file.

Once all the data has been calculated and collected, the overall evaluation of the weapon is calculated by multiplying all the values from the 3 arrays for each weapon together to determine the effectiveness rating for each weapon. If the weapon is the currently held weapon it is given a bonus so as to prevent circumstances were the evaluations of 2 weapons are very close and slight changes in situation cause constant changing between weapons. Each time the rating of a weapon is calculated, it is compared with the previous weapon and the one with the highest value is recorded. At the end of the calculations, the weapon with the highest rating is returned.

4.3.1. Weapon fuzzy data

The data that defines the fuzzy sets for the input variables, distance and the ammunition level, is integral to the performance of the fuzzy logic component. The values that were used to define the sets were carefully chosen after careful observation of a number of games, with collection of the information displayed on the screen for accurate appraisal.

Distance variable data definition

The set data for the distance variable is the same for all of the weapons; except for the gauntlet which is a special case in that it can only hit an opponent when in direct contact with it. The distance variable is split into 4 categories:

(i) close,
(ii) medium,
(iii) far,
(iv) very far.

It was decided that, with the exception of the gauntlet, each of the weapons generally could be rated according to the same ranges defined by the 4 categories and so just 2 sets of fuzzy sets needed to be used: one for the gauntlet and the other for the rest of the weapons.

The fact that the gauntlet needs to be touching the opponent to make a hit means that only 1 of its sets has to be considered when designating its min, max, and rule values; the *close* set. Anything outside the *close* set means that the weapon cannot damage the opponent and so is given the worst rule (bad). The major consideration was determining the "killing range" of the weapon, when it would be deemed usable. Table 1 shows the value ranges for the sets (far and very far not shown for readability as they also have rule of 0); minL and minH being the lower_min and upper_min values.

The gauntlet data shows that it is only usable within a distance of 5 units from the enemy. At any other distance, it will output the lowest value possible. This means that, combined with its low weapon weight, all other weapons should be chosen before it.

The set data for the other weapons needed to take into account the different ranges of the weapons, some being good at close range but almost useless at long range, others being good in the middle ranges but less so when close or very far away. It was decided that the upper range for distance was 1500 units, anything above this being set to 1500, as this was near the limit of the bot's awareness.

Figure 7 shows the data values used for each of the sets' ranges, the rules needed to be defined for each of the weapons separately as each has its different strengths and weaknesses (which can be seen in Table 2).

The data values chosen for each weapon were derived from observation of games being played and data collection from personal experience from playing the game. Access to the fuzzy logic component of the Q3A AI was not possible, so the values are a "best guess" as to the values used. The rules represent the weapon's ability to damage at each distance range, the lowest being 0 (bad) with the best being 3 (excellent).

Table 1: Distance fuzzy set data for gauntlet.

Close				Medium				Far				Very Far			
minL	minH	max	rule	minL	minH	max	rule	—	···	—	—	—	···	—	—
0	5	0	1	0	200	20	0	—	···	—	—	—	···	—	—

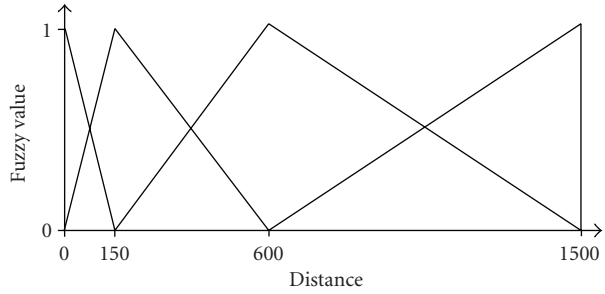


Figure 7: Fuzzy set data for weapon selection distance input.

Table 2: Fuzzy rule set for weapon selection distance.

	Rules			
	Close	Medium	Far	Very far
Machine Gun	1	1	1	1
Shotgun	3	2	1	0
Grenade launcher	1	2	2	1
Rocket launcher	1	3	2	2
Lightning gun	2	3	2	0
Railgun	1	2	3	3
Plasma gun	3	3	2	1
BFG	3	3	3	2

A number of factors were taken into account when defining the data values. For instance, the rocket launcher is given a *close* rule of 1 (average) because it has a large splash damage radius which will cause damage to the bot if used at close range. Its long range effectiveness is marked down due to the relatively slow speed of rockets which means that they are easy to avoid given the time that longer ranges afford. As another example, the railgun is only rated at 1 for close range due to its long reload time between shots, even though a single hit could kill the opponent.

Ammunition variable data definition

Each of the weapons needed to have its own set of data defined for the amount of ammunition input variable. All the weapons have different firing rates that determine the range for each of the ammunition sets. The only data that is constant between all the weapons, except for (again) the gauntlet which does not use ammunition at all, was the maximum amount of ammunition that could be available which is 200.

In the same way as was done for determining the values used for the distance sets, careful observation of the game lead to the selection of the set data, depending on the rate of fire of each weapon and the amount that is available when the weapon is first picked up. The amount of ammunition available on the map is not taken into account for this iteration of the project, although it could be a future improvement.

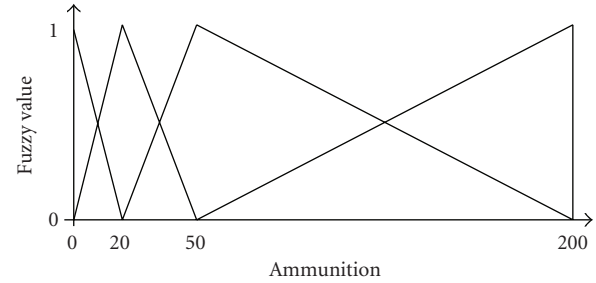


Figure 8: Fuzzy set data for shotgun ammunition.

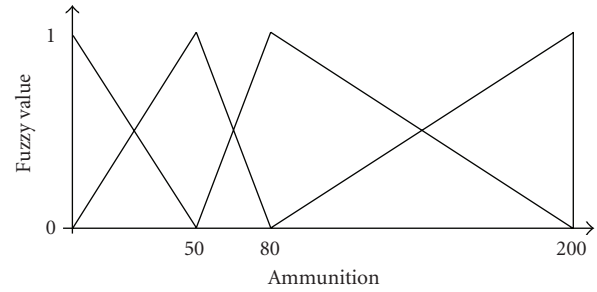


Figure 9: Fuzzy set data for plasma gun ammunition.

The set data for the ammunition for the shotgun and the plasma gun are shown in Figures 8 and 9. the diagrams show how the shotgun and plasma gun differ in the way they use ammunition. The shotgun has a slow reload and so the ranges of the sets are smaller and they are grouped near to 0 to represent the fact that, due to its slow firing rate, smaller amounts of ammunition are considered good. The plasma gun, in contrast, has a high firing rate which can be seen by the way the sets are more spaced out with larger ranges making larger amounts of ammunition more important than for the shotgun.

Output data definition

The output sets are defined by a single value that represents the max value of the set. This is the only value that needs to be specified due to the defuzzification method used; that of mean of maximum which only uses the max value to calculate the output value.

As can be seen in Figure 10, the output sets are not equally spaced from 0 to 100, each set is assigned a value to bias the output for that set. By doing this, the better rated sets produce much higher output values than the *bad* set, which can offset weapons with much higher characteristic

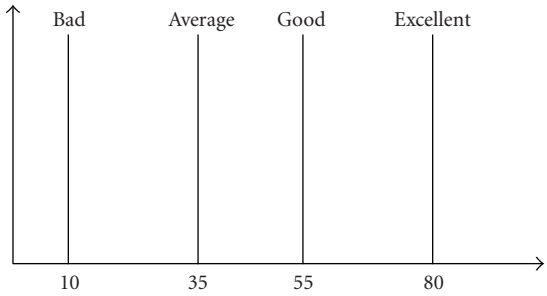


Figure 10: Fuzzy set data for weapon selection output.

```
EvalFuzzyWeapons()
        GetinputValues()
        for each weapon
                if have weapon and ammo
                        Fuzzify()
                        Composition()
                        Defuzzify()
                end of if
        end of for
        return array of weapon evaluations
end of function
```

Algorithm 6: Pseudocode for weapon selection fuzzy logic evaluation function.

preferences. This was done so that there would be very little chance of selecting weapons in circumstances were they are useless, such as when the enemy is out of range and so no damage can be inflicted.

4.3.2. Weapon selection fuzzy logic evaluation function

This function is called from the *SelectBestWeapon()* function in order to get the fitness values for all the weapons according to the fuzzy logic component. From this function, the three parts of the fuzzy logic component are called, as can be seen in Algorithm 6.

First, the input values for the inputs are extracted and then, if the bot has the weapon in its inventory and also has ammunition for the weapon, the fuzzy logic component is run on them (calling the three parts in turn—*Fuzzify, Composition* and *Defuzzify*). An array is returned at the end that holds the evaluation of all the weapons by the fuzzy logic component.

4.3.3. Weapon selection perceptron evaluation function

The function that evaluates the weapons using the perceptron component simply uses the code explained in Section 4.2. For the weapon selection behaviour, like the fuzzy logic evaluation function, the evaluation is only run if the weapon is in the bot's inventory and there is an ammunition available for it.

The output of the perceptron is normalised to a value between 0 and 1 by dividing it by the number of inputs into the perceptron. Each of the inputs is normalised also and by taking the maximum number of inputs that can be used at one time, the perceptron output can be scaled appropriately. The maximum number of inputs takes into account the categorisation of some inputs, which means that the four inputs resulting from categorisation actually represent 1 input as, at most, there will only be two active at one time and their combined values will always be approximately 1.

Perceptron inputs for weapon selection

The inputs used for the perceptron are the following:

(i) distance to enemy: categorised to 4 inputs;
(ii) ammunition: categorised to 4 inputs;
(iii) health;
(iv) visibility: categorised into 2 inputs, visible or not visible;
(v) height difference: categorised into 2 inputs, above or below;
(vi) aggression.

The inputs used were determined from assessment of those specified in the analysis section during development. Some of the input variables proposed were discovered to be superfluous to the adaptation process and others had to be tailored to the limitations of the Q3A source code. The quad input was discarded due to the test map not including the power-up, although this could be implemented if other maps were to be used. The visibility input was originally intended to be a measure of obstructions in the area but it was discovered that the function that returns the visibility of the enemy only returns the values 0 and 1 on the test map (although on maps that include fog it returns values between 0 and 1). This meant that it is now only used to determine whether the bot can see its opponent or not and is used choosing weapons that have splash damage that can still hit the enemy even when they are hidden.

The visibility and the height difference inputs are categorised simply into 2 inputs, each being either a 1 or 0 depending upon whether they are visible or not or whether the enemy is above or below. If in one category that input value is 1 and the other 0 and vice versa. This enables the perceptron to gain positive or negative reinforcement for situations when the enemy is not visible (1 in not visible input) or when they are (1 in visible input).

Fuzzy set data for input categorisation

The categorisation of the inputs distance and ammunition are needed to scale the inputs,so that each weapon would have the same output from the perceptron for the same relative inputs. This presented no problems for the distance input as the distance to the enemy is the same for all the weapons. The ammunition input required the amounts to be scaled appropriately for each of the weapons individually so as to represent the characteristics of each, and give

roughly the same perceptron output for the same relative level of ammunition. In order to do this, the input set data used for the ammunition by the fuzzy logic component was used.

4.3.4. Feedback for weapon selection

The feedback for the weapon selection behaviour required careful thought on how to deal with the varying characteristics of the different weapons. The feedback needed to be a measure of the weapon that dealt the most damage to the enemy, whilst also taking into account any damage done to the bot. This meant that a timing element needed to be introduced to account for the different firing rates of the weapons. Another consideration was that some projectiles take time to travel to their target whilst others strike immediately. The weapons fall into one of two categories that affect the feedback.

(i) Instant shot: those weapons whose projectiles impact immediately upon firing (having a speed of 0). Weapons in this category are
   (a) gauntlet (W_1),
   (b) machine gun (W_2),
   (c) shotgun (W_3),
   (d) lightning gun (W_4),
   (e) railgun (W_5).

(ii) Missile: weapons that fire projectiles that have a finite speed and take time to impact. Weapons in this category are
   (a) grenade launcher (W_6),
   (b) rocket launcher (W_7),
   (c) plasma gun (W_8),
   (d) BFG (W_9).

The method developed to record the input data (distance, ammunition, etc.) when a firing event occurs (when the weapon is fired). When the projectile impacts, hitting either opponent or environment, it calls the feedback function which determines the time between impacts and the damage inflicted on the enemy and sustained by the bot. This deals with the problems of calling feedback just after a fire event (would not know damage for missiles) or calling feedback when the missile impacts (need to know inputs when projectile fired). To implement this method, each fired projectile needs to be tracked after it is fired, so that it can be related to the correct inputs when it impacts. The time between impacts is only measured when in combat with the enemy, not including time between combat when navigating the map. The previous impact time is reset each time the bot finds a new enemy.

Based on the requirements of the feedback, (6) (weapon selection feedback equation) shows how the feedback value is calculated from the damage to the enemy, the damage to the bot, the time between projectile impacts and a bonus. The bonus is given when the enemy is killed by the current weapon:

$$\text{feedback} = \frac{\text{damage2enemy} - \text{damage2self}}{\text{last\_impact\_time} - \text{prev\_impact\_time}} + \text{bonus}. \tag{6}$$

The feedback value is a representation of the damage/second inflicted by the weapon. In order to be able to directly compare this value with that of the perceptron output value for the training it needs to be normalised to the range [0, 1].

Normalisation of weapon selection feedback

The normalisation of the feedback requires that it represent how well the weapon is performing. Every time the weapon is fired it produces feedback whether it hits or not. To fairly judge the performance of the weapons, the characteristics of each weapon was analysed so that a good measure of a good performance could be established. Table 3 shows the firing characteristics of each weapon.

Using this data, the average damage per second, taking into account all the weapons, was estimated to be 150 (rounding down to the nearest 10). This figure assumes that the weapons are 100% accurate. The average accuracy level of a "standard" bot, at skill level 4, was calculated to be approximately 25%. Taking this into account, the average damage the weapons inflict could be estimated at 150/4 = 37.5. This value could then be used as an "average" feedback score, one which should produce a normalised value of 0.5.

In keeping with observed accuracy, most of the time the weapon is going to miss, on average hitting only 1 in 4 times. This means that when training the perceptron with the missed shots, the reduction in value of the weights inflicted by misses should be compensated for when the enemy is hit inducing damage. The time between shots also has a bearing on the performance measure, but an average damage per second of 37.5, taking into account misses and hits, should train the perceptron to output 0.5 in those situations.

The upper and lower limits of the feedback value could also be estimated from the weapon data, using the max splash damage to determine the lower value and max damage per second to determine the upper. After investigating the source code it was discovered that splash damage only inflicts 0.5 of the damage on the attacker, so the minimum level is 50 (100*0.5 with no hit on enemy). The maximum damage per second is capped at 250 due to the excessive damage of the BFG, which is available only on a few maps and usually has little ammunition available for it.

In order to adequately normalise the feedback to give a value that could be used for training of the perceptron, fuzzy logic is used. This enables input values to output specific values, used to output 0.5 from an input of 37.5, and replaces what could be a mathematically complex function with a simple process.

Figure 11 shows the input fuzzy sets that are used to normalise the feedback value. The 4 sets span the range of values from −50 to 250 that the feedback falls between with set 1 centred on 0, the feedback for a miss, and set 2 centred on

Table 3: Weapon firing characteristics.

W#	Damage/projectile	Speed of projectile (0 = instant)	Splash damage	Splash radius	Fire delay (1/10 sec)	Damage/second
W_1	50	0	0	0	400	125
W_2	7	0	0	0	100	70
W_3	90	0	0	0	1000	90
W_4	100	700	100	150	800	125
W_5	100	900	100	120	800	125
W_6	16	0	0	0	200	80
W_7	100	0	0	0	1500	67
W_8	20	2000	15	20	100	200
W_9	100	2000	100	120	200	500

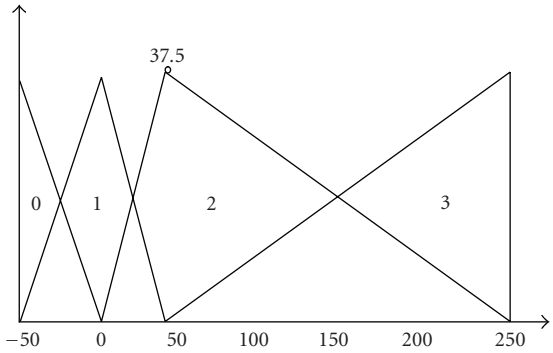


Figure 11: Feedback normalisation input fuzzy set data.

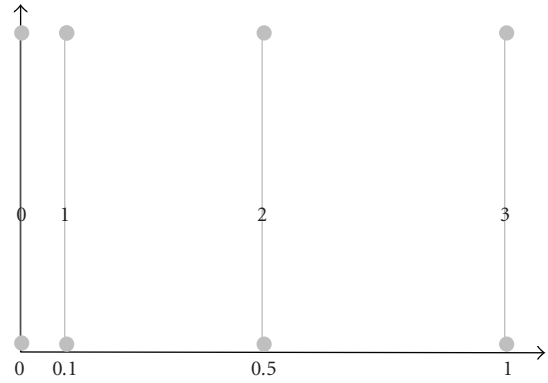


Figure 12: Feedback normalisation output fuzzy set data.

37.5, the feedback for an average hit. Each of the sets directly map to the output sets, shown in Figure 12, by way of their rule.

The output sets are set up so that they output specific values for specific input values. An input value of 37.5 will have a 100% membership in set 2 which maps to the output set 2, producing an output of 0.5. An input of 0 will have 100% membership in set 1, mapping to output set 1 giving an output of 0.1. Inputs of −50, the minimum feedback, and 250, the maximum, will output 0 and 1, respectively.

```
weaponFeedback()
  last_impact_time = current_time
  damage_enemy = damage to enemy
  damage_self = damage to self from splash from own
weapon
  if enemy killed
    bonus = 80
end of if
feedback_value = (damage_enemy − damage_self)/
                 (last_fire_time − prev_fire_time) + bonus
feedback_value = normaliseFeedback(feedback_value)
 trainPerceptron(feedback_value)
 prev_impact_time = last_impact_time
 last_impact_time = 0
end of function
```

Algorithm 7: Pseudocode for weapon selection feedback function.

Pseudocode for weapon selection feedback

The feedback function is called after every projectile impact, extracting the information required and then calling the function that trains the perceptron. Algorithm 7 shows pseudocode for the operation of the function.

First, the time is recorded so that the time since the previous impact occurred can be calculated. The damages done to the enemy and self are extracted and a bonus given if the enemy died from this projectile. The feedback value is then calculated and normalised before being passed to the perceptron training function to adjust the weights. Finally, the previous impact time is made equal to the current time and the last impact time is set to 0.

4.3.5. *Training for weapon selection*

The training of the perceptron is simply involved using the delta rule algorithm to update the weights of the perceptron based on the output of the perceptron when the projectile was fired and the feedback gathered from the projectiles impact.

```
trainPerceptron()
        for each input to perceptron (i)
            weight[i] + = learning_rate*
                        (feedback_value − output_value)*
                          input_value[i]
        end of for
end of function
```

Algorithm 8: Pseudocode for weapon selection training function.

Pseudocode for weapon selection training function

The function for training the perceptron is quite simple in operation as can be seen in Algorithm 8. The learning rate is set at a low value, which will be altered during evaluation in order to find a good balance between adapting fast enough to influence a game and slow enough so that isolated events do not interfere with the appropriate learning.

The function simply loops through the inputs to the perceptron, calculating the change required to the weight by multiplying the difference between the perceptron output and the feedback by the learning rate and the input value. This adjustment is then added to the weight, increasing or decreasing its value accordingly.

4.4. System development

The design section discussed the methods that are used to implement the adaptation system. This section will show how the designs for the components were realised using the Q3A engine [8, 15].

The adaptation system is composed of a number of functions relating to a specific component or behaviour within the system. All of the functions that were specially written for this system use the same prefix "*adapt*" in order to easily find and identify them within the many functions that make up the Q3A source code. The majority of the adaptation system functions and data structures are defined in the *ai_main.h* and *ai_main.c* files of Q3A engine [8, 15]. This provides access to and from the Q3A AI functions and data structures that this system integrates with.

The functions developed and a description of their purpose are listed below.

(i) *Fuzzy logic component functions:*

  (a) *AdaptFuzzify* fuzzifies the input data;
  (b) *AdaptComposition* calculates the degree of membership for the output sets;
  (c) *AdaptDefuzzify* calculates the output of the fuzzy logic component.

(ii) *Weapon selection behaviour:*

  (a) *AdaptSelectBestWeapon* is a main function that evaluates the weapons and selects the best available.

(iii) *Weapon selection behaviour fuzzy logic functions:*

  (a) *AdaptLoadWeaponFuzzyVals* loads the weapon fuzzy values from the characteristic files;
  (b) *AdaptAllocWepDOMS* loads the fuzzy set data;
  (c) *AdaptEvalFuzzyWeapons* main function that is called to evaluate the weapons using fuzzy logic.

(iv) *Weapon selection behaviour perceptron functions:*

  (a) *AdaptInitWPercept* initialises the perceptron weights and loads the set data for categorisation of inputs;
  (b) *AdaptEvalPerceptWeapons* is a main function that is called to evaluate the weapons using the perceptron;
  (c) *AdaptGetWinputvalues* extracts the data for the perceptron inputs;
  (d) *AdaptWeaponFeedback* calculates the feedback for the weapon selection behaviour;
  (e) *AdaptTrainWPerceptron* trains the perceptron based on the feedback.

(v) *Utility functions:*

  (a) *AdaptFlagFireEvent* records a firing event;
  (b) *AdaptOutputWepEvaluation* outputs the perceptron data to file.

4.5. Fuzzy logic component development

4.5.1. Data structures

The fuzzy logic component was required to fulfil a number of roles in the system: evaluation of actions, categorisation of inputs to the perceptron, and normalisation of values. In order to meet these requirements, two data structures were created to contain the information used to calculate the output from the fuzzy logic component.

(i) *adapt_DOM_t*: this structure contains a 2-dimensional array to hold the data for each set of an input variable.
(ii) *adapt_FL_t*: this structure contains an array of adapt_DOM_t structures for each input variable and an array containing the output set data. It also has variables for the number of sets and the number of input variables.

By splitting the 2 structures up, it allowed the use of a single adapt_DOM_t structure for categorisation of inputs, in which case the output set is not needed.

4.5.2. Loading fuzzy set data

The set data for the fuzzy logic component comprises a large amount of information; each input variable having a number of sets each requiring 4 values to define it (MINL, MINH, MAX, and RULE). For the weapon-selection behaviour, this meant 9 weapons each with 2 inputs each of which had 4 sets defined by 4 values, resulting in 288 (+4 for output set) values that needed storing. In order to facilitate the use of this

```
0 20 0 1 0 200 20 0 20 1500 200 0 200 1500 1500 0
0 10 0 0 5 100 50 0 50 150 100 0 100 200 200 0
0 150 0 1 0 600 150 2 150 1500 600 1 600 1500 1500 1
0 50 0 1 0 100 50 1 50 200 100 2 100 200 200 2
...
```

Algorithm 9: Extract from fuzzy set data file for weapon selection.

```
line_start = 0
read file and put all data into input_string
for each weapon
        for each input variable
            while newline character not encountered in
        input_string
                        line_length++
            end of while
           copy data between line_start and line_length to
        line_string
        scan line_string for values and put them into
        adapt_FL_t structure
          line_start + = line_length+1
          line_length = 0
         end of for
end of for
```

Algorithm 10: Pseudocode for function to parse fuzzy set data.

amount of data, it is loaded in from a text file on initialisation of the game.

The data is stored in a particular format, as shown in Algorithm 9. The structure of the data can be seen in Table 4 (data missing for clarity); the first two lines of the data representing the input variables for a single weapon, each line representing all the data for each input. A line is made up of four groups of four data items, representing the data for each fuzzy set in ascending order.

In order to load the data, the function *AdaptAllocWepDOMS* needed to be written which extracted the data from the file and put it into the appropriate place in the *adapt_FL_t* structure used to hold the information. The file utilities incorporated into QuakeC [15] are basic, much of which is tailored to specific purposes within the game engine, especially when parsing the loaded data.

The Q3A *trap_FS_Read()* function enables data to read in from the file and be placed into an array of characters. The data then needs to be parsed and placed into the data structure. Algorithm 10 shows the pseudocode for how the data is parsed from the string read in from file.

The function sets up loops for each weapon and each input variable and searches the string for a newline character. The data from the start of the line to the end of the line is then copied to another string, using the Q3A *strncpy()* function, and the Q3A function *sscanf()* is then used to scan through the copied string for the individual data values contained within. Finally, the variable holding the position of the start of the line is set to the start of the next line and the new line length set to 0.

4.5.3. Loading weapon characteristic data

Part of the requirements of the system is that it takes into account the fuzzy preferences that are defined in the characteristic files. Access to the fuzzy component of Q3A is not available in the source code and the data loaded into it could not be extracted for use by the adaptable AI system. This meant that a function needed to be written to load the data from the characteristic file, so that the weapon weights could be used.

The weapon preference file is formatted in a particular way. In a similar way to that of the fuzzy set data, the data needed to be parsed so that the values could be placed into an array. The process of extracting the data required a different technique, as the values are linked to a key representing the weapon that the value applies to. Algorithm 11 shows the pseudocode for parsing the weapon preference data from the file string.

The function goes through the string, character by character, looking for prefix of a key, "*W_*." When it finds this combination of characters, it finds the end of the line and copies that line to another string which is scanned for the key and a value. The key is compared with the weapon names and when a match is found, the value is placed in an array at the appropriate location for that weapon (the weapon number).

4.6. Weapon selection perceptron component development

4.6.1. Data structure

The simple architecture of the perceptron means that the data structure required to hold the data for it is also simple. The structure created, called *adapt_P_t*, is made up of two arrays and an integer variable. The variable simply holds the number of inputs into the perceptron, for use when looping through perceptrons with varying numbers of inputs. Due to the limitations in allocating memory imposed by the Q3A engine, the arrays need to set up to the size of the largest number of inputs into the perceptron. Therefore, the arrays can be of any size so the *numinputs* variable is used when determining the size of the arrays.

One array is used to store the perceptron weights for each of the inputs. The other array stores the input values for the last time the output of the perceptron was calculated. This is required for the training of the perceptron, which needs the value for each input in order to calculate the adjustment for the weights.

4.7. Weapon selection behaviour development

The weapon selection behaviour is controlled through a single function that calls the separate components, calculates the best weapon, and returns the weapon number. It replaces the Q3A function *trap_BotChooseBestFightWeapon()* in the *BotChooseWeapon()* function for the adaptable bot. The function is called *AdaptSelectBestWeapon()*.

Table 4: File format for weapon selection fuzzy set data.

		Set 1				···	···	Set 4		
Weapon	Input	MINL	MINH	MAX	RULE	···	···	MINH	MAX	RULE
Gauntlet	Distance	0	20	0	1	···	···	1500	1500	0
	Ammo	0	10	0	0	···	···	200	200	0
MachGun	Distance	0	150	0	1	···	···	1500	1500	1
	Ammo	0	50	0	1	···	···	200	200	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

```
while not at end of input_string
        if input_string[character_num] = "W" and
          input_string[character_num+1] = "–"
                line_start = character_num+2
                while newline character not encountered
                        line_length++
                end of while
                copy data between line_start and
          line_length to line_string
                scan line_string for key and value
                if key = weapon_name
                        array[weapon] = value
                end of if
        end of if
        character_num++
end of while
```

Algorithm 11: Pseudocode for function to parse weapon preference data.

4.7.1. Information gathering

The weapon selection process requires a number of details about the bot, its enemy and the environment to be gathered. This is done by making use of the data structures and functions defined in Q3A. The distance to the enemy and the ammunition level are common to both the fuzzy logic and perceptron components. Both of these pieces of information can be found in the *inventory* array from the *bot_state_t* data structure. This array contains a large amount of useful information, the ones used for the weapon selection being as follows.

(i) *bs->inventory[INVENTORY_MACHINEGUN]* returns a 1 if the bot currently has the weapon in its inventory. The names of each weapon can be substituted for MACHINEGUN.

(ii) *bs->inventory[INVENTORY_BULLETS]* returns the amount of ammunition type. The names of each type of ammunition can be substituted for BULLETS.

(iii) *bs->inventory[INVENTORY_ARMOR]* returns the amount of armour the bot currently has.

(iv) *bs->inventory[ENEMY_HEIGHT]* returns the difference in height between the bot and its current enemy.

(v) *bs->inventory[ENEMY_HORIZONTAL_DISTANCE]* returns the distance (horizontally) between the bot and its enemy.

Other variables within the *bot_state_t* structure also contain information used for the inputs to the perceptron. The health level of the bot is found using

$$\text{bs- > lastframe\_health} \tag{7}$$

while it can be determined if the bot is directly in combat with another bot or player using

$$\text{bs- > enemy} \tag{8}$$

which returns a -1 if not in combat and the entity number of the enemy when it is.

In order to find the visibility of the enemy and the aggression of the bot, functions need to be called that return the level of each. To get the visibility of the enemy, the function

$$\text{BotEntityVisible()} \tag{9}$$

is called. This returns a floating point number in the range [0, 1], although values other than 0 or 1 are only returned when in fog or water. Otherwise, the value returned simple represents whether the enemy is visible or not. The aggression level of the bot is found using

$$\text{trap\_Characteristic\_BFloat()} \tag{10}$$

and passing CHARACTERISTIC_AGGRESSION as a parameter. This returns a floating point number in the range [0, 1] giving the aggression value defined in the characteristic file.

```
if weapon fired
search through wep_fire_event array for empty slot (i)
if it is not allocated
        weapon_fire = i
        wep_fire_event[i] = event_type
end of if
```

Algorithm 12: Pseudocode for flag fire event function.

4.7.2. Projectile tracking for feedback

The majority of the code for this application was confined to the AI sections of the engine as it purely deals with the behaviour of the bots within the game. Due to the way the feedback is calculated, recording information after a shot and when the projectile impacts, the weapons needed to signal when they fired and the projectile emitted from the weapon needed to be tracked until it impacted on either the environment or the enemy.

In order to achieve this, two different sections of the game engine, the AI (files prefixed with "*ai_*") and the game sections (files prefixed with "*g_*"), were required to communicate with each other using a common data structure that was available to both. The *playerState_t* data structure seemed to provide the answer as it was accessible from *bot_state_t*, which the AI used, and from *gclient_s* which was accessible from the game section. This proved to be problematic in practice due, what appeared to be the same structure in fact being different versions of one another so, data stored in one version from the game section would be copied to the one available to the AI section, but changes made in the AI section were not available from the game section.

This meant changing the location of the data from *player_State_t* to *gclient_t*, which was accessible to the AI section by the way of a global structure that makes data available across the whole server side of the game engine.

When a weapon is fired, the projectile is tracked using an array stored in *gclient_t*, as shown in Algorithm 12. The *FlagFireEvent()* function is in the file *g_weapon.c* and is called from the functions that fire the weapons, also in the same file.

The projectile entity, if a missile, stores the array location of the event, in order that it can be identified upon impact, and stores the type of event (missile or impact) in the *wep_fire_event* array. The missile event is used for the delayed impacts of rockets, grenades, plasma, and the impact event is used for instant shot projectiles and when the missile projectiles impact. An array is used for cases when multiple missiles are active at the same time, for instance when the plasma gun fires a volley or the grenade launcher launches a number of grenades at once. The missiles can impact in any order so the array must search for unallocated slots.

When a missile entity impacts, calling either the *G_MissileImpact()* or *G_ExplodeMissile* functions in *g_missile.c*, it sets the event in the wep_fire_event array to impact and resets the weapon_fire variable. At the end of an AI cycle, the wep_fire_event array is checked for impact events and if one is found, the feedback function is called, resetting the wep_fire_event array to free the slot for other projectiles. It is also checked for missile events and, if a new event is found, a copy of the perceptron input values are put into a wep_event_inputs array that is stored in the *bot_state_t* structure.

5. EVALUATION

For the purposes of testing the adaptation system, the fuzzy logic component was designed to mimic the selections made by the original Q3A AI as closely as possible. This was done so that the changes in behaviour of the bot due to adaptation during a match could be directly compared with the behaviour exhibited by the original AI.

5.1. Adaptation of weapon selection

In order to test whether the bot is able to change weapon preferences within the game, its preferences were set up so that it had a high preference for a certain weapon but also low accuracy. By assigning another weapon a high accuracy but normal preference, the system's ability to change preferences was tested. The adaptable bot's preferences and accuracy levels were set up as follows:

(i) *plasma gun:* accuracy = 0.1, preference = 300;
(ii) *rocket launcher:* accuracy = 0.9, preference = 200;
(iii) *grenade launcher:* accuracy = 0.8, preference = 100;
(iv) *shotgun:* accuracy = 0.7, preference = 150.

The significant weapon numbers in Figures 6 and 7 are 2-machine gun, 3-shotgun, 4-grenade launcher, 5-rocket launcher, and 8-plasma gun.

Figure 13 shows the output of the weapon selection choices, comparing the Q3A AI with the adaptable AI. The graph shows how the adaptable AI and Q3A AI make very similar selections at the start of the match, with only slight variations in the choice of weapon. Towards the end of the match the differences of choice become more evident with regards to the plasma gun in particular; seen clearly in Figure 14 which shows a close up of the last part of the match.

This demonstrates the adaptation occurring on the plasma gun's use as, due to its very low accuracy, the negative feedback lowers its effectiveness rating over the course of the match until the other weapons effectiveness scores make them a better choice. The rating of the plasma gun falls so low that the grenade launcher, with only a third of the preference rating of the plasma gun, is preferred over it in some situations. The rocket launcher is shown to be preferred to the plasma gun in almost all situations, and those times when it is not can be accounted for by the rocket launcher running out of ammunition.

The graphs showing the adaptation of the perceptrons for the plasma gun (Figure 15) and the rocket launcher (Figure 16) illustrate how rating of the plasma gun drops and the rocket launcher rises to a point were the preferences change for the weapons. Whereas, the plasma gun's *medium* range drops to around 0.2, the rocket launcher's rises, albeit

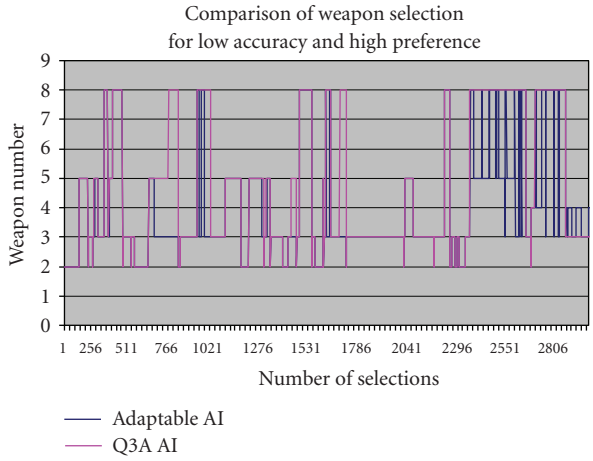


Figure 13: Graph of weapon selection comparison between adaptable ai and q3a ai for low accuracy and high preference of plasma gun.

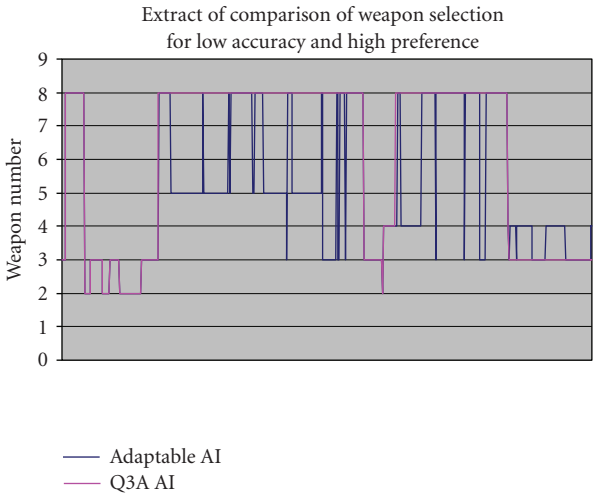


Figure 14: Extract of comparison of weapon selection for low accuracy and high preference, showing difference due to adaptation.

only slightly, to 0.55. This is enough of a variation to cause the change in weapon selection to occur.

5.2. Validity of input choices for perceptron

The inputs chosen for the perceptron resulted in varying degrees of success in their ability to affect the selection of weapons due to adaptation. The distance input was successful in reflecting the feedback of the weapon's strengths and weaknesses in the adaptation of its weights. Trends can be identified from the adjustments made during training that relate to the performance of the weapon in the game.

Another input that demonstrated an effect on the selection process was the height difference, although not to the extent of the distance input. It showed a higher effectiveness for when the enemy is below the bot and lower for enemies above.

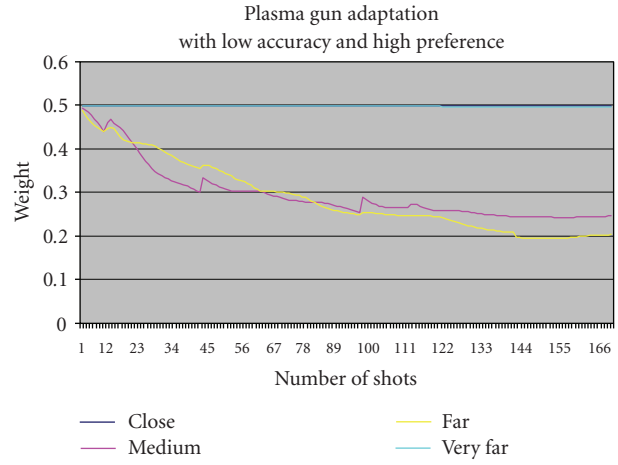


Figure 15: Adaptation of plasma gun distance due to low accuracy and high preference.

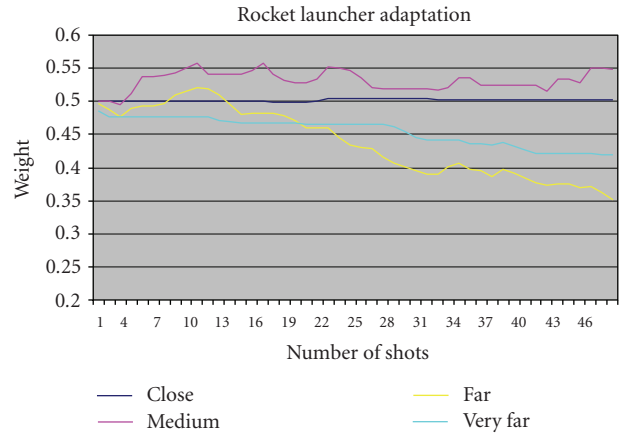


Figure 16: Adaptation of rocket launcher distance due to high accuracy and medium preference.

The ammunition input showed little influence over determining the correct weapons by adapting its values. This is because there is no direct link between the effectiveness of the weapon and the amount of ammunition, therefore the feedback could not influence the ammunition training. The only direct influence of the ammunition on the weapon selection came when the level fell to 0, causing the weapon to be changed to another. All other levels had no bearing on how the weapon performed, indicating that categorisation was not required. Possibly, restricting the ammunition input to a "low ammunition" input would better serve the selection of weapons.

The evaluation of the inputs shows that certain types of input lead to better performance of the adaptation of the perceptron while others contribute little. Generally, the most effective inputs:

(i) directly influence the behaviour; the ammunition input had no direct influence over the effectiveness of the weapon, whereas the distance changed how well it performed;

(ii) are reflected in the feedback; there was no feedback that reflected the effect of ammunition at levels other than 0, when the weapon needed to change. The amount of damage that could be inflicted was not affected by larger or smaller amounts of ammunition.

6. CONCLUSIONS

This paper has shown how, by combining traditional AI techniques, a system can be developed that enables the choices made by a conventional AI layer to be altered in response to feedback from the actions selected. Although the development was limited to just the weapon selection behaviour, so limiting the effect on the game that is visible from testing, evidence was found that the system is capable of adapting to feedback by a significant enough amount to change the actions that prove unsuccessful to those that are successful. The results showed interesting trends that indicate that, with more development and testing to determine optimum settings, the system developed could form the basis of a useable adaptable AI system.

ACKNOWLEDGMENT

The authors would like to express their appreciation to the anonymous reviewers for their very helpful and constructive comments and recommendations.

REFERENCES

[1] J. Manslow, *Learning and Adaptation*, AI Game Programming Wisdom, Charles River Media, Rockland, Mass, USA, 2002.
[2] J. E. Laird, "Game AI: the state of the industry 2000, part two," *Game Developer Magazine*, 2000.
[3] M. McCuskey, *Fuzzy Logic for Video Games*, Game Programming Gems, Charles River Media, Rockland, Mass, USA, 2000.
[4] S. Russell and P. Norvig, *Artificial Intelligence 'A Modern Approach'*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.
[5] S. Rabin, *AI Programming Wisdom*, Charles River Media, Rockland, Mass, USA, 2002.
[6] N. Palmer, "Machine Learning in Games Development," 2003, http://ai-depot.com/GameAI/Learning.html.
[7] A. J. Champandard, *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviours*, New Riders, Indianapolis, Indiana, 2004.
[8] J. M. P. van Waveren, "The Quake III Arena Bot," University of Technology Delft, Faculty ITS, San Diego, Calif, USA, 2003.
[9] T. Alexander, *An Optimized Fuzzy Logic Architecture for Decision Making*, AI Game Programming Wisdom, Charles River Media, Rockland, Mass, USA, 2002.
[10] M. Zarozinski, *Imploding Combinatorial Explosion in a Fuzzy System*, Game Programming Gems 2, Charles River Media, Rockland, Mass, USA, 2001.
[11] R. C. Berkan and S. L. Trubatch, *Fuzzy Systems Design Principles: Building Fuzzy IF-THEN Rule Bases*, Wiley-IEEE, New York, NY, USA, 1997.
[12] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice-Hall, Upper Saddle River, NJ, USA, 1994.
[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 1998.
[14] M. R. Medsker and J. And Liebowitz, *Design and Development of Expert Systems and Neural Computing*, Macmillan College, New York, NY, USA, 1994.
[15] Quake 3 Arena, Id Software, Dallas, Tex, USA, 1999, http://www.idsoftware.com.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 281959, 9 pages
doi:10.1155/2008/281959

Research Article

Generation of Variations on Theme Music Based on Impressions of Story Scenes Considering Human's Feeling of Music and Stories

Kenkichi Ishizuka and Takehisa Onisawa

Graduate School of Systems and Information Engineering, University of Tsukuba, Tennodai 1-1-1, Tsukuba 305-8573, Japan

Correspondence should be addressed to Takehisa Onisawa, onisawa@iit.tsukuba.ac.jp

Received 31 July 2007; Accepted 17 October 2007

Recommended by Kevin Kok Wai Wong

This paper describes a system which generates variations on theme music fitting to story scenes represented by texts and/or pictures. Inputs to the present system are original theme music and numerical information on given story scenes. The present system varies melodies, tempos, tones, tonalities, and accompaniments of given theme music based on impressions of story scenes. Genetic algorithms (GAs) using modular neural network (MNN) models as fitness functions are applied to music generation in order to reflect user's feeling of music and stories. The present system adjusts MNN models for each user on line. This paper also describes the evaluation experiments to confirm whether the generated variations on theme music reflect impressions of story scenes appropriately or not.

1. INTRODUCTION

Music, pictures, and/or text information are combined into multimedia content with interaction among them [1]. The effectiveness of multimodal communication using combined different modal media has been analyzed in the field of cognitive psychology [1]. It is expected that multimodal communication will be performed in everyday life in the future owing to the development of information technology [2]. However, the interaction among different modal media is not necessarily generated by their simple and random combination. Features and impressions of media should be considered well in order to create effective multimedia contents. Therefore, creation of multimodal contents costs more time and labor than that of single-modal one. Support systems for creation of multimodal contents or for the flexible combination of different modal media are taken interest in [3, 4].

The authors are studying on the construction of a system which generates variations on theme music fitting to each story scene represented by texts and/or pictures [5]. This system varies melodies, tempos, tones, tonalities, and accompaniments of a given theme music based on impressions of story scenes. This system has two sections representing (a) relations between story scenes and musical images and (b) relations between features of variations and musical impressions. Since human feeling of stories and music is different among people [6] and the difference is important in multimedia content creation, it is necessary to consider the above relations depending on each user. Although in [5] these relations are obtained by questionnaire data, that is, off line, in the present paper a method, which adjusts the relations for each user on line, is proposed. In this paper, the transformation of theme music is defined as follows. Tunes, tones, musical performances, rhythms, tempos are varied according to story scenes [7].

2. OUTLINE OF PRESENT SYSTEM

2.1. Inputs and outputs

Inputs to the present system are original theme music and numerical information on given story scenes. Outputs are MIDI files of variations on original theme music generated according to each story scene. This paper deals with generation of variations on theme music fitting to stories obtained by the system [8] that generates story-like linguistic

Table 1: Information on story scene.

No.	Information	Num	Information
1	Happiness	7	Kind of character 1
2	Sadness	8	Kind of character 2
3	Surprise	9	Impressions of behavior
4	Fear	10	Kind of character 1 (previous)
5	Anger	11	Kind of character 2 (previous)
6	Disgust	12	Impressions of behavior
—	—	13	Picture's sequence

expressions given four pictures. In this paper, a scene is defined as each picture for story generation. Information on a picture scene, for example, character's emotion, kinds of characters, impressions of character's behavior in a story scene (e.g., violent behavior), picture sequence, as shown in Table 1, is acquired from each picture [8]. These are inputs to the present system.

2.2. System structure

The present system consists of two sections, a musical image acquisition (MIA) section and a theme music transformation (TMT) section as shown in Figure 1. The MIA section converts information on story scenes shown in Table 1 into transformation image parameters (TIPs) by modular neural network (MNN) models [9]. The TMT section transforms inputted original theme music based on values of TIPs, and generates a set of midiformatted candidates of variations on theme music for each story scene. The TMT section applies genetic algorithms (GAs) to the generation of variations candidates, which has MNN models as fitness functions. MNN models consist of three neural network models, an average model network (AMN), an individual variation model network (IVMN), and gating networks. AMN is a hierarchical neural network model expressing user's average feeling of music and stories. IVMN is a radial basis function network model expressing differences among users' feeling of music and stories. The gating network switches over between AMN and IVMN. The present system adjusts IVMNs and the gating networks for each user.

3. MUSICAL IMAGE ACQUISITION (MIA) SECTION

The MIA section is constructed by MNN models. The inputs to MNN models are shown in Table 1. MNN models estimate the values of TIPs representing musical image for transformation of original theme music. In this paper, TIPs consist of some pairs of adjectives that are selected referring to a study that retrieves many genres musical works with pairs of adjectives representing musical image [10]. These are *happy-sad*, *heavy-light*, *hard-soft*, *stable-unstable*, *clear-muddy*, *calm-violent*, *smooth-rough*, *thick-thin*. Preexperiments are performed in order to confirm which pairs of adjectives are necessary for TIPs. The procedures of the preexperiments are as follows. (1) Fixing musical instruments, tempos, tonalities, tones, chords in a melody part and accompaniment parts patterns at random, 125 variations are generated. (2) Some subjects, who have no experience to play some musical instruments over 3 years, listen to the variations and express impressions on them with 8 pairs of adjectives. (3) If the subjects feel that it is difficult to evaluate the difference among the variations with some pairs of adjectives, they give the pairs. The results of the pre-experiments show that it is difficult to evaluate the difference among the variations using adjectives *hard-soft*, *stable-unstable*, *smooth-rough*, or *thick-thin*. Then, in this paper these four pairs of adjectives are not used. That is, four pairs of adjectives, which are parameters on *degree of change from original theme music* shown in Table 2, are used. Each parameter value is a real number in [0.0, 1.0].

Table 2: Transformation image parameters.

Parameters	Values
Calm-violent	[0.0–1.0]
Heavy-light	[0.0–1.0]
Happy-sad	[0.0–1.0]
Clear-muddy	[0.0–1.0]
Degree of change from original theme music	[0.0–1.0]

The MIA section estimates the values of TIPs from information on a picture scene. In generation of variations on theme music fitting to story scenes, information on story scenes necessary for the estimation of the values of TIPs is dependent on media representing a story, for example, pictures, texts or animations or the contents of a story, for example, a serious story, a story for children, and is not determined uniquely. Therefore, it is necessary to consider the selection of information on picture scenes for the estimation of the values of TIPs. However, since in this paper, input to the present system is limited to information on pictures scenes, the paper does not discuss this point. In the future it is necessary to change information according to media representing a story or the form of a story.

4. THEME MUSIC TRANSFORMATION (TMT) SECTION

4.1. Procedure on generation of variations [5]

Inputs to the TMT section are original theme music and values of TIPs obtained by the MIA section, and outputs are MIDI files of variations on theme music. MIDI files consist of the melody part and six accompaniment parts. The accompaniment parts consist of an obbligati part, a backing parts 1 and 2, a bass part, a pad part, and a drums part. The TMT section modifies impressions of inputted original theme music varying the following components of MIDI files [5]: (1) scores of melody parts, (2) tempos, (3) tonalities, (4) accompaniment patterns of accompaniment parts, and (5) tones.

4.2. Structure of TMT section

The TMT section transforms given original theme music according to inputted TIPs and outputted sets of MIDI-formatted candidates of variations on given theme music as shown in Figure 2. GAs are applied to the transformation of

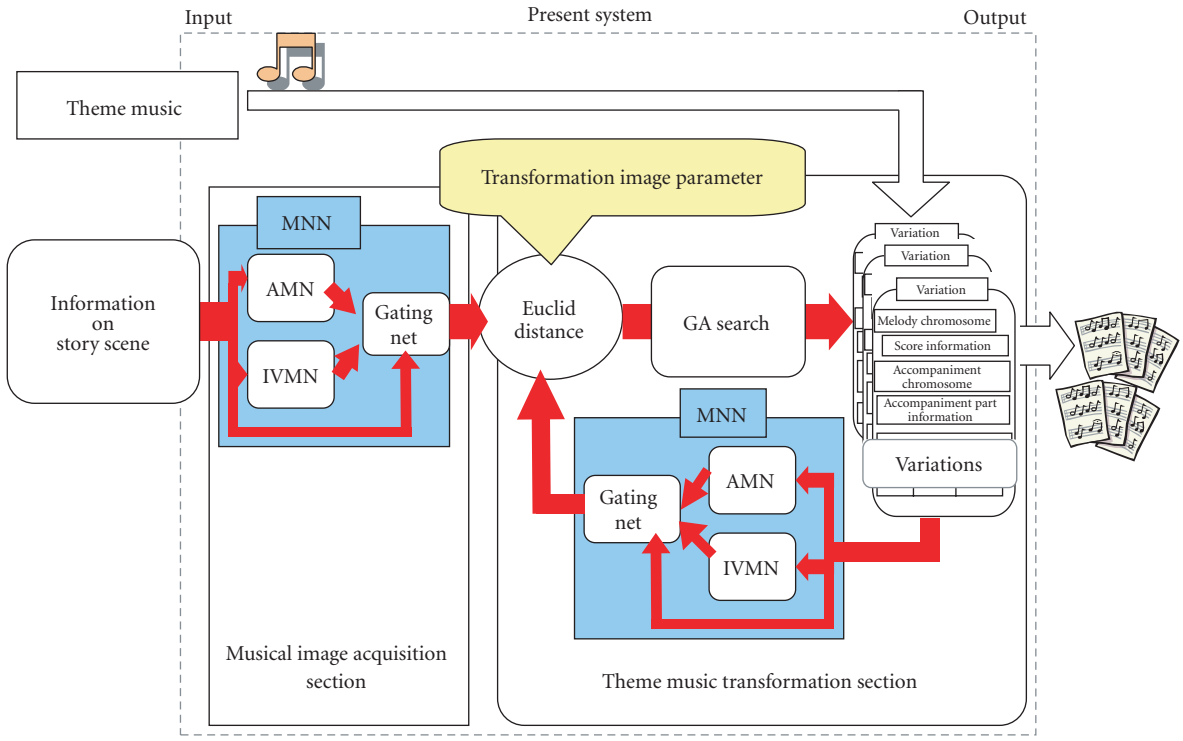


Figure 1: System structure.

a given theme music fitting to TIPs, where a variation generated from a given theme music is represented by a chromosome in the framework of GAs. In this paper, GAs parameters are abbreviated as follows.

(1) N: Population size
(2) T: Maximum number of generations
(3) N_{new}: The number of individuals generated randomly
(4) N_{user}: Partial population size presented to user
(5) P_c: Crossover probability
(6) P_m: Mutation probability.

Procedures in the TMT section are as follows.

(1) N variations are generated from inputted theme music in the form of chromosomes.
(2) Fitness values of chromosomes are calculated according to the inputted values of TIPs and melodies of original theme music.
(3) GAs operations of crossover and mutations are performed. Next generation population is generated. Go back to step (2).

4.2.1. Structure of chromosome

Variations consist of three kinds of chromosomes such as *Melody Chromosome*, *Accompaniment Chromosome*, and *Status Chromosome*.

The melody chromosome has melody part score information. Melody part score information is represented by the format shown in Figure 3. A given original theme music is represented as an initial chromosome. The accompaniment chromosome has accompaniment part information. The playing pattern number and the performance type of the obbligati part in the accompaniment part are represented by chromosomes, where each information is represented with 1 byte as shown in Figure 3. Initial chromosomes have random values for information. The status chromosome has information on a tempo, a tonality, and a tone. Tempo, tonality, melody part tone, and obbligato part tone are represented by a chromosome as shown in Figure 3. Tempo (60–200 [BPM]), tonality (a major scale or minor one), and tone are also represented with 1 byte. Initial chromosomes have random values for information.

4.2.2. Calculation of fitness value [5]

Fitness values of chromosomes are calculated according to the inputted values of TIPs and melodies of original theme music [5]. Let i ($i = 1, 2, \ldots, N$) be the chromosome number, that is, the variation number, and Fitness_i represents the fitness value of the ith variation. Fitness_i is defined as

$$\text{Fitness}_i = \text{Melody Fitness}_i + \text{Impression Fitness}_i, \quad (1)$$

where Melody Fitness_i is the fitness value of score information in the melody part of the ith variation referring to [11], and $\text{Impression Fitness}_i$ is the fitness value of impressions on the ith variation [5]. Impression values of variations are estimated by MNN models. These impression values are degrees of four pairs of adjectives used in TIPs estimation. MNN models are obtained by the relation between feature spaces of variations and impression values.

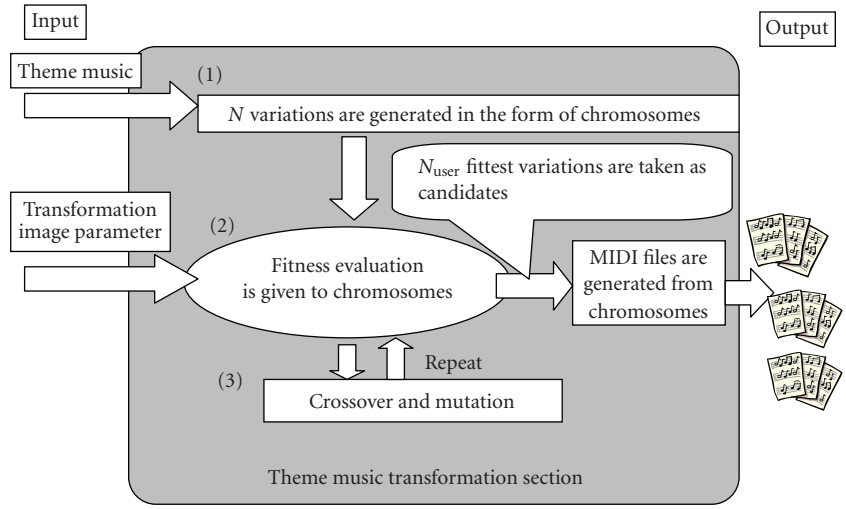


Figure 2: Theme music transformation section.

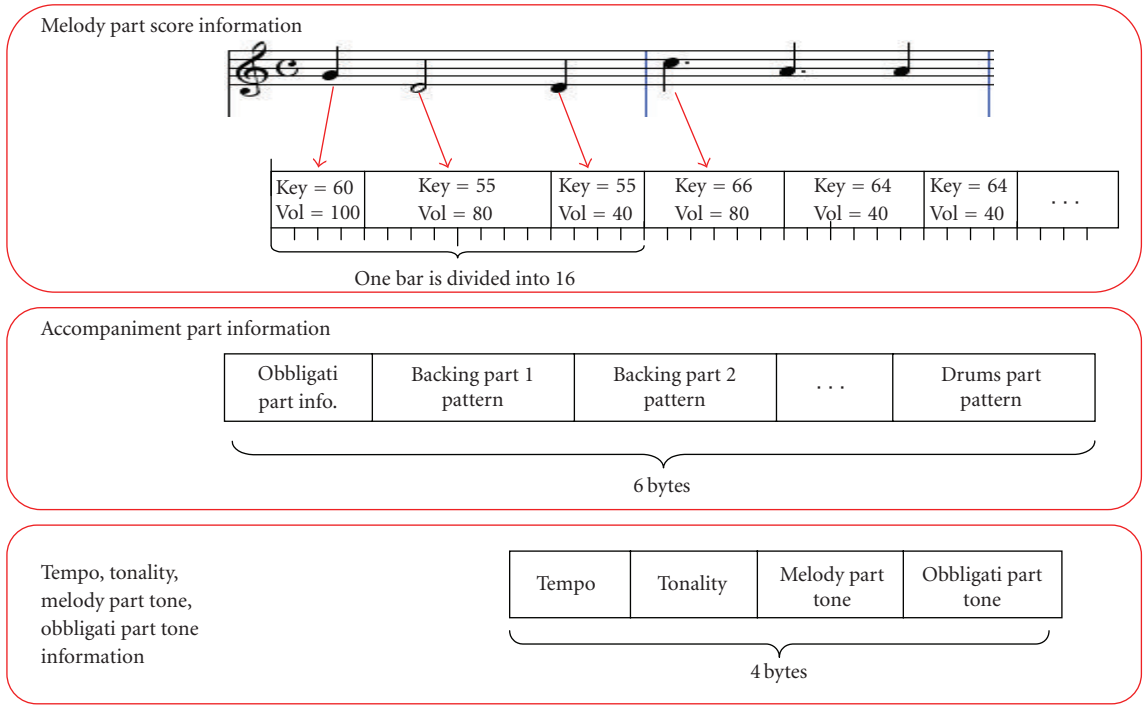


Figure 3: Three kinds of chromosomes.

Smaller the value of Fitness_i is, the better the ith variation is. Procedures of calculation of fitness values are shown in Figure 4.

4.2.3. GA operations

$(N - N_{\text{new}})$ individuals of parent candidates are selected by the tournament selection according to the fitness values obtained in 4.2.2. Crossovers at probability of P_c and mutations at P_m are applied to parent candidates. N_{new} individuals are generated at random. Crossover and mutation are performed as follows.

Crossover

uniform crossover is applied to melody chromosomes obtained by the generative theory of total music grouping structure analysis [12] in every group.

Mutation

random values are assigned to the accompaniment chromosome and the status chromosome. Varying score information on the melody part described in [5] is applied to melody chromosomes.

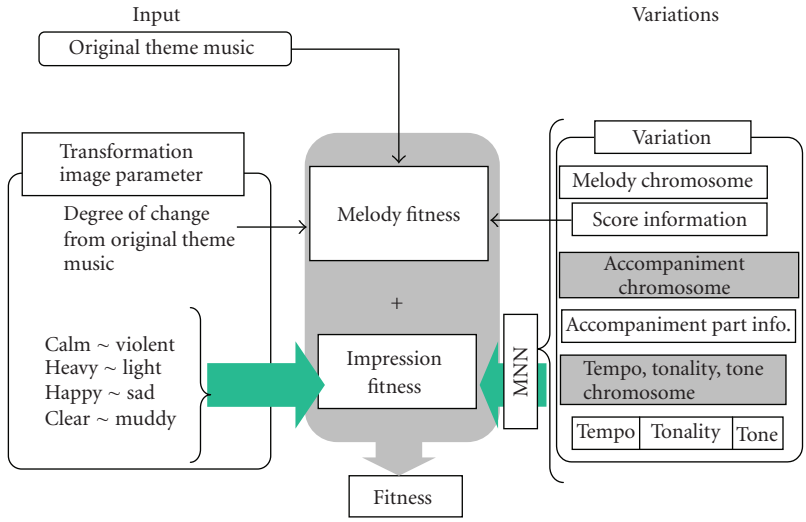


Figure 4: Calculation of fitness values.

5. MNN STRUCTURE

The present system uses MNN models to represent (1) relations between story scenes and values of TIPs in the MIA section, and (2) relations between features of variations and musical impressions in the TMT section. MNN models in the present system consist of AMN, IVMN, and the gating network as shown in Figure 5. When the present system adjusts its MNN models for each user, IVMN and the gating network are obtained by learning of user's data of individual variation of feeling of music and stories.

AMN is a hierarchical neural network model which consists of sigmoid neurons. AMN is constructed using questionnaire data of subject's feelings for music and stories. The questionnaire data are obtained referring to [6].

IVMN is a hierarchical neural network model which consists of RBF neurons. RBF is a function responding to input values in a local area. Therefore, an RBF network is easy to be adjusted online and fast. When a user is not satisfied with outputs of MNN, learning data of IVMN are generated and saved in the present system. Input values of learning data are input values of MNN. Output values of learning data are evaluation values by each user.

The gating network is an RBF network switching over between AMN and IVMN. The gating network judges whether input values of MNN are close to the area learned by IVMN or not. When a user is not satisfied with outputs of MNN, learning data of IVMN are generated and saved in the present system. Learning data of the gating network are input values of MNN. Output values of learning data of the IVMN are evaluation values by users.

IVMN and the gating network are constructed by the method proposed in [13] using all data saved in the present system.

Outputs of MNN models are defined as

$$f_{\mathrm{MNN}}(x) = \begin{cases} f_{\mathrm{personal}}(x) : g(x) \geq t \\ f_{\mathrm{average}}(x) : g(x) < t, \end{cases} \quad (2)$$

where $g(x)$ is an output value of the gating network $f_{\mathrm{personal}}(x)$ is an output value of the IVMN $f_{\mathrm{average}}(x)$ is an output value of the AMN, and t is a threshold of switching AMN and IVMN.

6. EXPERIMENTS

Experiments are performed to evaluate the present system by 8 undergraduate/graduate students. In the experiments, GA parameters are set at the following values: $N = 100$, $T = 100$, $N_{\mathrm{user}} = 3$, $N_{\mathrm{new}} = 20$, $P_c = 70\%$, $P_m = 20\%$. In the experiments, the threshold of switching AMN and IVMN by a gating network is set at 0.75. Musical works are chosen at random from prepared seventeen MIDI files of classical tunes or folk tunes, and are used as theme music of stories.

6.1. Construction of IVMN and gating network

IVMN and the gating network for each subject are constructed in the following procedures.

(1) Story scenes and theme music are inputted to the present system. The present system generates N variations according to each story scene and outputs them.
(2) When a subject is satisfied with one of outputted variations, go to (8). When a subject is not satisfied with any variations, go to (3).
(3) A subject looks at the values of TIPs estimated by the present system. The values of TIPs are presented to a subject in the form of Figure 6.
(4) The present system adjusts MNN models according to two cases as shown in Figure 7. That is, a subject feels (a) presented musical image is not suitable for story scenes or (b) generated variations are different from presented musical images.
    (a) When a subject feels that presented musical image is not suitable for story scenes, a subject evaluates whether the values of TIPs fit to story

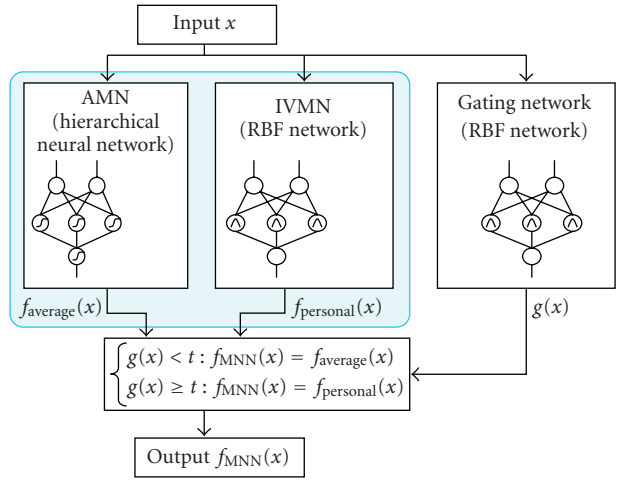


Figure 5: Concept figure on MNN.

scenes by the interface shown in Figure 6. Variations are generated according to values of TIPs evaluated by a subject. Go to (5).

(b) When a subject feels generated variations are different from presented images, a subject chooses one of$N$ variations. A subject evaluates his/her impressions using the 7-point scale method shown in Figure 8, where evaluation items are pairs of the same adjectives as the ones used as TIPs estimation. In this procedure the human interface shown in Figure 8 is used. Variations are generated by using modified MNN models. Go to (6).

(5) A subject listens to N variations. When a subject is satisfied with one of outputted variations, go to (7). When a subject is not satisfied with any variations, go to (3).

(6) A subject listens to N variations. When a subject is satisfied with one of outputted variations, go to (8). When a subject is not satisfied with any variations, go to (3).

(7) MNN models in MIA section are adjusted by the relation between information on story scenes and values of TIPs evaluated by a subject.

(8) Go to an evaluation of the next scene.

6.2. Experiment 1

Three story scenes are inputted into the present system and variations are generated, where the story scenes are different from the ones used in Section 6.1 and MNN models are adjusted for each subject in Section 6.1. This experiment confirms whether the present system generates variations on theme music reflecting subject's feeling of music and stories.

Let twelve story scenes be $S_i (i = 1, \ldots, 12)$. A subject is asked to read S_i and to evaluate musical images fitting S_i using the 7-point scale method (e.g., (7) very calm through (1) very violent), where evaluation items are 4 pairs of the same adjectives as the ones used in TIPs estimation. Let the evaluation values of S_i by a subject be $I_{S_i} = (a_{i1}, a_{i2}, a_{i3}, a_{i4})$ and let twelve variations generated from S_i by the present system be $P_i (i = 1, \ldots, 12)$. A subject is asked to evaluate impressions of P_i by 7-point scale method, where evaluation items are four pairs of the same adjectives as the ones used in TIPs estimation, and P_i are presented to a subject at random. Let impressions of P_i evaluated by a subject be $I_{P_i} = (b_{i1}, b_{i2}, b_{i3}, b_{i4})$, where a_{i1} and b_{i1}, a_{i2} and b_{i2}, a_{i3} and b_{i3}, and a_{i4} and b_{i4} are evaluation values of "violent-calm," "heavy-light," "clear-muddy," and "sad-happy," respectively. These variables have integer values in [1, 7] evaluated by a subject. In this experiment, cosine correlations [14] between I_{S_i} and I_{P_i} are used for the evaluation whether the generated variations are reflecting subject's feelings for music and stories or not. Cosine correlation $\text{Sim}(I_{S_i}, I_{P_i})$ is defined as

$$\begin{aligned} \text{Sim}(I_{S_i}, I_{P_i}) &= \cos\left(\arg\left(I_{S_i}, I_{P_i}\right)\right) \\ &= \frac{\sum (a_{ij} \times b_{ij})}{\sqrt{\sum (a_{ij})^2} \times \sqrt{(b_{ij})^2}}, \quad (1 \leq j \leq 4), \end{aligned} \tag{3}$$

when $\text{Sim}(I_{S_i}, I_{P_i})$ is close to 1.0, generated variations are reflecting users feelings for music and story well.

6.3. Result 1

$\text{Sim}(I_{S_i}, I_{P_i})$ are shown in Table 3. It is found that 80% of the whole of $\text{Sim}(I_{S_i}, I_{P_i})$ is 0.9 or more, and the present system is able to generate variations reflecting subject's feelings to music and stories.

6.4. Experiment 2

Other three story scenes are inputted into the present system and variations are generated, where MNN models in the present system are adjusted for each subject in Section 6.1. A subject is asked to evaluate with 7-point scale method whether variations on theme music fit impressions of each presented story scene or not; (7) very suitable (6) suitable (5)

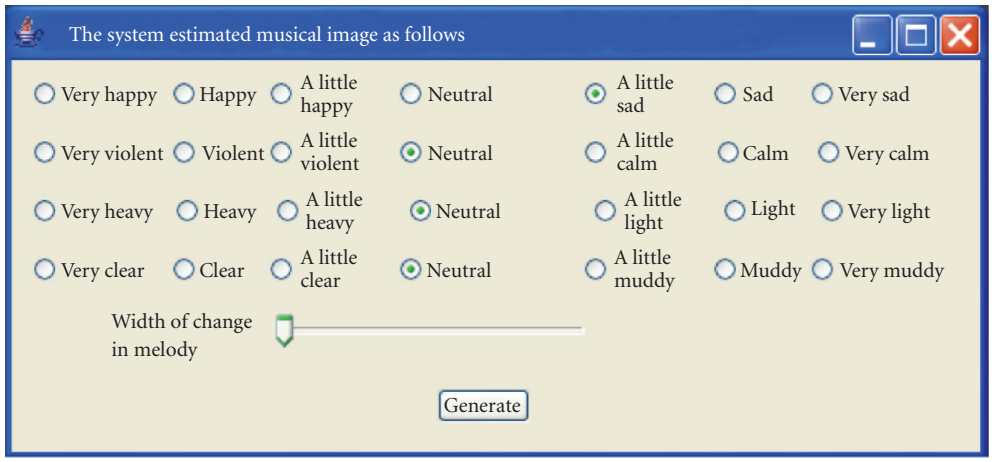


Figure 6: TIPs estimation by present system.

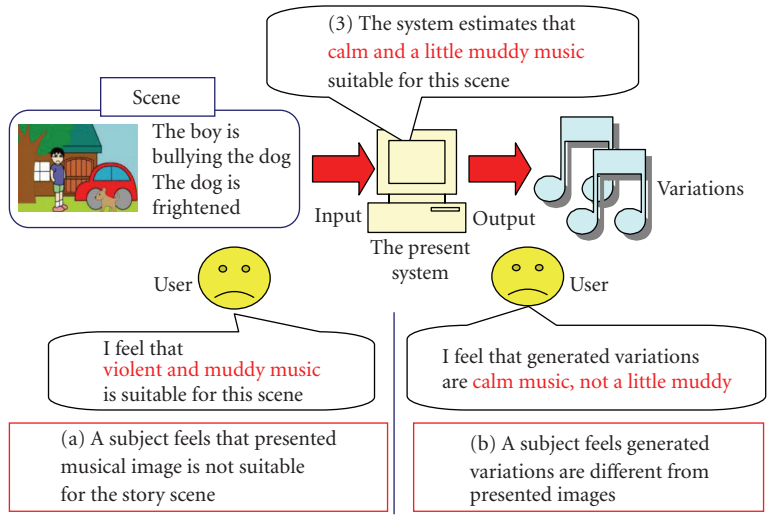


Figure 7: MNN models adjustment.

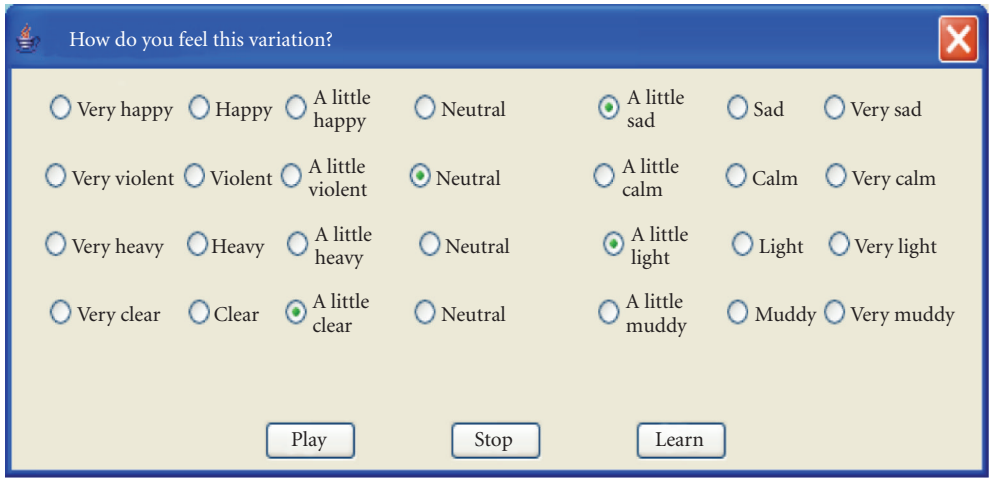
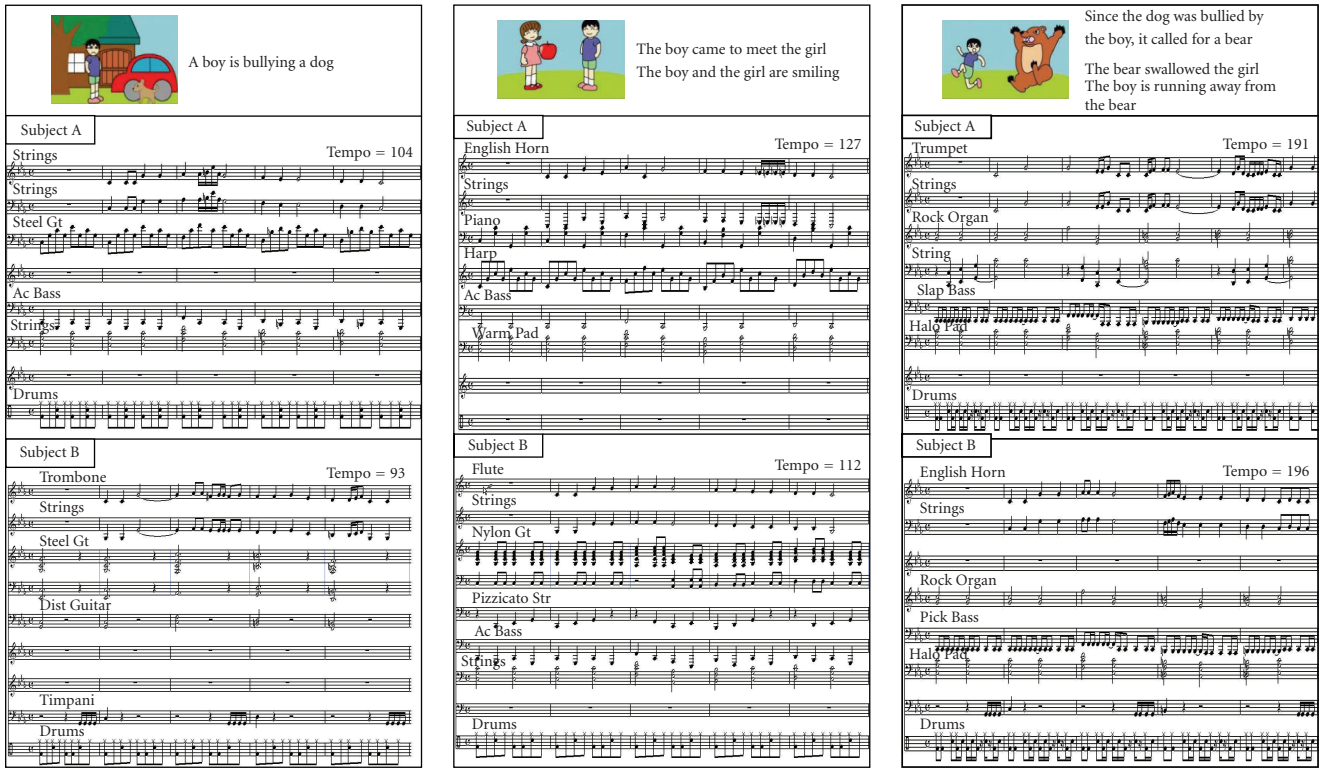


Figure 8: Interface.

Graph	Average of evaluation values	Decentralization of evaluation values
Average system	4.97	1.82
Present system	5.69	1.06

Figure 9: Distributions of evaluation values.



(a) (b) (c)

Figure 10: Examples of variations.

a little suitable (4) neutral (3) little suitable (2) not suitable (1) not suitable at all. Furthermore, to confirm the effectiveness of IVMN, variations generated by the present system and the ones generated by an average system are compared with each other, where the average system is the system whose IVMN in MNN models are average among subjects.

6.5. Result 2

Experimental results are shown in Table 4. It is found that the present system gets evaluation (5) or (6) or (7) in approximately 87.5% of all evaluation results.

Distributions of evaluation values of variations generated by the present system and those of variations generated by the average system are shown in Figure 9. It is found that variations generated by the present system are evaluated higher in the large percentage than variations generated by the average system. An average and a decentralization of evaluation values are also shown in Figure 9. The decentralization of evaluation values of variations generated by the present system is lower than that of variations generated by the average system. It is found that constructing IVMN and the gating network for each user are effective.

Figure 11: Theme music.

Figure 10 shows examples of the variations on theme music by subjects A and B, where original theme music is Twinkle Stars as shown in Figure 11. From these figures it is found

Table 3: Cosine Similarity.

Story	Scene	Subject							
		G	H	I	I	K	L	M	N
1	1	0.86	0.94	0.93	0.95	0.96	0.98	0.99	0.97
	2	0.83	0.99	0.98	0.97	0.98	0.99	0.98	0.97
	3	0.94	0.99	0.82	0.95	0.95	0.99	0.98	0.94
	4	0.96	0.90	0.96	0.96	0.96	0.95	0.98	0.94
2	1	0.84	0.97	0.97	0.94	0.94	0.95	0.94	0.89
	2	0.91	0.97	0.99	0.85	0.99	0.99	0.99	0.98
	3	0.98	0.95	0.98	0.93	0.93	0.97	0.98	0.95
	4	0.97	0.92	0.96	0.82	0.91	0.97	1.00	1.00
3	1	0.84	0.94	0.83	0.95	0.85	0.98	0.98	0.97
	2	1.00	0.98	0.99	0.96	0.98	0.99	0.99	0.99
	3	0.99	0.93	0.98	0.94	0.95	0.93	0.88	0.40
	4	0.65	0.98	0.79	0.80	0.98	0.97	0.94	0.50
	Average	0.90	0.96	0.93	0.92	0.95	0.97	0.97	0.88

Table 4: Evaluation Values.

Story	Scene	Evaluation Values							
		G	H	I	I	K	L	M	N
1	1	6	6	5	4	5	6	3	5
	2	6	7	7	6	5	5	5	4
	3	5	7	7	6	6	7	6	6
	4	6	6	6	7	3	7	6	4
2	1	7	7	4	6	5	7	4	6
	2	6	5	6	5	6	6	6	5
	3	6	6	7	6	6	7	6	5
	4	7	6	7	7	4	7	6	6
3	1	7	7	6	5	5	5	4	5
	2	7	6	5	6	6	6	4	5
	3	5	6	6	7	4	5	6	5
	4	7	6	6	6	4	6	5	3

that although the same scenes are given to the subjects, various theme tunes are transformed by the present system.

Variations on theme music generated by the present system are dependent on subjects' impressions on story expressed by pictures. Therefore, even if the same pictures are given, generated variations are different among subjects. Nevertheless, subjects themselves are satisfied with generated variations. Then it is found that the present system generates variations on theme music fitting to subjects' impressions on story well. However, subjects' impressions on story usually change according to time and environment in which subjects are. The present system does not deal with the variations depending on these factors, time, environment, and so forth. This is a future work.

7. CONCLUSIONS

This paper presents the system which transforms a theme music fitting to story scenes represented by texts and/or pictures, and generates variations on the theme music. The present system varies (1) melodies, (2) tempos, (3) tones, (4) tonalities, and (5) accompaniments of a given theme music based on impressions of story scenes using neural network models and GAs. Differences of human's feeling of music/stories are important in multimedia content creation. This paper proposes the method that adjusts the models in the present system for each user. The results of the experiments show that the system transforms a theme music reflecting user's impressions of story scenes.

REFERENCES

[1] Z. Iwamiya, *Multimodal Communication on Music and Visualizations*, Kyushu University Press, Fukuoka, Japan, 2000.

[2] S. Takahasi, M. Okamoto, and H. Ohara, "Voice and sound processing technology for easy, comfortable, convenient communications environment," *NTT Technical Journal*, pp. 8–9, 2004 (Japanese).

[3] H. Liu, H. Lieberman, and T. Selker, "A model of textual affect sensing using real-world knowledge," in *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03)*, pp. 125–132, Miami, Fla, USA, January 2003.

[4] H. Takagi and T. Noda, "Media converter with impression preservation using a neuro-genetic approach," *International Journal of Hybrid Intelligent Systems*, vol. 1, no. 1, pp. 49–56, 2004.

[5] K. Ishizuka and T. Onisawa, "Generation of variations on theme music based on impressions of story scenes," in *Proceedings of the International Conference on Games Research and Development*, pp. 129–136, Perth, Western Australia, December 2006.

[6] Y. Kiyoki, T. Kitagawa, and T. Hayama, "A metadatabase system for semantic image search by a mathematical model of meaning," *ACM SIGMOD Record*, vol. 23, no. 4, pp. 34–41, 1994.

[7] W. Apel, *Harvard Dictionary of Music*, Harvard University Press, London, UK, 2nd edition, 1973.

[8] S. Kato and T. Onisawa, "Generation of consistent linguistic expressions of pictures," *Journal of Japan Society for Fuzzy Theory and Intelligent Infomatics*, vol. 17, no. 2, pp. 233–242, 2005 (Japanese).

[9] K. Watanabe, *Neural Network Computational Intelligence*, Morikita Press, Tokyo, Japan, 2006.

[10] T. Ikezoe, Y. Kazikawa, and Y. Nomura, "Music database retrieval system with sensitivity words using music sensitivity space," *Journal of Japan Information Processing Society*, vol. 42, no. 12, pp. 3201–3212, 2001 (Japanese).

[11] T. Kadota, M. Hirao, A. Ishino, M. Takeda, A. Shinohara, and F. Matsuo, "Musical sequence comparison for melodic and rhythmic similarities," in *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE '012001)*, pp. 111–122, Laguna De San Rafael, Chile, November 2001.

[12] F. Lerdahl and R. Jackendoff, *A Generative Theory pf Tonal Music*, MIT Press, Cambridge, Mass, USA, 1983.

[13] A. Hattori and H. Nakayama, "Additional learning and active forgetting by support vector machine and RBF networks," Tech. Rep., Institute of Electronics, Information and Communication Engineers, Tokyo, Japan, 2002.

[14] A. Yamaue and S. Kurachi, *Psychological Statistics*, Kitaoozi Press, Tokyo, Japan, 1991.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 412056, 7 pages
doi:10.1155/2008/412056

Research Article

A Constraint-Based Approach to Visual Speech for a Mexican-Spanish Talking Head

Oscar Martinez Lazalde, Steve Maddock, and Michael Meredith

Department of Computer Science, Faculty of Engineering, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK

Correspondence should be addressed to Oscar Martinez Lazalde, acp03om@sheffield.ac.uk

Received 30 September 2007; Accepted 21 December 2007

Recommended by Kok Wai Wong

A common approach to produce visual speech is to interpolate the parameters describing a sequence of mouth shapes, known as visemes, where a viseme corresponds to a phoneme in an utterance. The interpolation process must consider the issue of context-dependent shape, or coarticulation, in order to produce realistic-looking speech. We describe an approach to such pose-based interpolation that deals with coarticulation using a constraint-based technique. This is demonstrated using a Mexican-Spanish talking head, which can vary its speed of talking and produce coarticulation effects.

1. INTRODUCTION

Film, computer games, and anthropometric interfaces need facial animation, of which a key component is visual speech. Approaches to producing this animation include pose-based interpolation, concatenation of dynamic units, and physically based modeling (see [1] for a review). Our approach is based on pose-based interpolation, where the parameters describing a sequence of facial postures are interpolated to produce animation. For general facial animation, this approach gives artists close control over the final result; and for visual speech, it fits easily with the phoneme-based approach to producing speech. However, it is important that the interpolation process produces the effects observed in the natural visual speech. Instead of treating the pose-based approach as a purely parametric interpolation, we base the interpolation on a system of constraints on the shape and movement of the visible parts of the articulatory system (i.e., lips, teeth/jaw, and tongue).

In the typical approach to producing visual speech, the speech is first broken into a sequence of phonemes (with timing), then these are matched to their equivalent visemes (where a viseme is the shape and position of the articulatory system at its visual extent for a particular phoneme in the target language, e.g., the lips would be set in a pouted and rounded position for the /u/ in "b**oo**"), and then intermediate poses are produced using parametric interpolation. With less than sixty phonemes needed in English, which can be mapped onto fewer visemes since, for example, the bilabial plosives /p/, /b/, and the bilabial nasal /m/ are visually the same (as the tongue cannot be seen in these visemes), the general technique is low on data requirements. Of course, extra postures would be required for further facial postures such as expressions or eyebrow movements.

To produce good visual speech, the interpolation process must cater for the effect known as coarticulation [2], essentially context-dependent shape. As an example of forward coarticulation, the lips will round in anticipation of pronouncing the /u/ of the word "stew," thus affecting the articulatory gestures for "s" and "t." The de facto approach used in visual speech synthesis to model coarticulation is to use dominance curves [3]. However, this approach has a number of problems (see [4] for a detailed discussion), perhaps the most fundamental of which is that it does not address the issues that cause coarticulation.

Coarticulation is potentially due to both a mental planning activity and the physical constraints of the articulatory system. We may plan to over- or underarticulate, and we may try to, say, speak fast, with the result that the articulators cannot realize their ideal target positions. Our approach

tries to capture the essence of this. We use a constraint-based approach to visual speech (first proposed in [4, 5]), which is based on Witkin and Kass's work on physics-based articulated body motion [6]. In [7], we presented the basics of our approach. Here, we show how it can be used to produce controllable visual speech effects, whilst varying the speed of speech.

Section 2 will present an overview of the constraint-based approach. Sections 3, 4, and 5 demonstrate how the approach is used to create Mexican-Spanish visual speech for a synthetic 3D head. Section 3 outlines the required input data and observations for the constraint-based approach. Section 4 describes the complete system. Section 5 shows the results from a synthetic talking head. Finally, Section 6 presents conclusions.

2. CONSTRAINT-BASED VISUAL SPEECH

A posture (viseme) for a phoneme is variable within and between speakers. It is affected by context (the so-called coarticulation effect), as well as by such things as mood and tiredness. This variability needs to be encoded within the model. Thus, a viseme is regarded as a distribution around an ideal target. The aim is to hit the target, but the realization is that most average speakers do not achieve this. Highly deformable visemes, such as an open mouthed /a/, are regarded as having larger distributions than closed-lip shapes, such as /m/. Each distribution is regarded as a constraint which must be satisfied by any final speech trajectory. As long as the trajectory stays within the limits of each viseme, it is regarded as acceptable, and infinite variety within acceptable limits is possible.

To prevent the ideal targets from being met by the trajectory, other constraints must be present. For example, a global constraint can be used to limit the acceleration and deceleration of a trajectory. In practice, the global constraint and the distribution (or range) constraints produce an equilibrium, where they are both satisfied. Variations can be used to give different trajectories. For example, low values of the global constraint (together with relaxed range constraints) could be used to simulate underarticulation (e.g., mumbling). In addition, a weighting factor can be introduced to change the importance of a particular viseme relative to others.

Using the constraints and the weights, an optimization function is used to create a trajectory that tries to pass close to the center of each viseme. Figure 1 gives a conceptual view of this. We believe that this approach better matches the mental and physical activity that produces the coarticulation effect, thus leading to better visual speech. In using a constrained optimization approach [8], we need two parts: an objective function $\mathrm{Obj}(X)$ and a set of bounded constraints C_j,

$$\text{minimize } \mathrm{Obj}(X) \quad \text{subject to } \forall j : \underline{b}_j \le C_j(X) \le \overline{b}_j, \quad (1)$$

where $\underline{b}_j$ and $\overline{b}_j$ are the lower and upper bounds, respectively. The objective function specifies the goodness of the system state X for each step in an iterative optimization procedure. The constraints maintain the physicality of the motion.

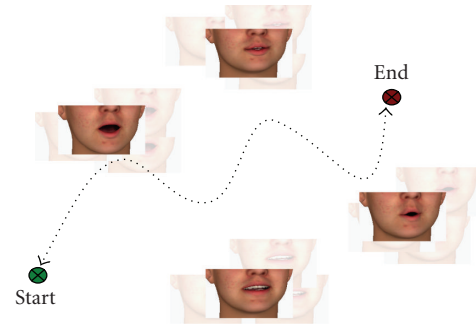


Figure 1: Conceptual view of the interpolation process through or near to clusters of acceptable mouth shapes for each viseme.

Table 1: Boundary constraints.

Constraints	Action
$S(t_{\text{start}}) = \varepsilon_{\text{start}}$	Ensures trajectory starts at $\varepsilon_{\text{start}}$
$S(t_{\text{end}}) = \varepsilon_{\text{end}}$	Ensures trajectory ends at ε_{end}
$S(t_{\text{start}})' = S(t_{\text{end}})' = 0$	Ensures the velocity is equal to zero at the beginning and end of the trajectory
$S(t_{\text{start}})'' = S(t_{\text{end}})'' = 0$	Ensures the acceleration is equal to zero at the beginning and end of the trajectory

The following mathematics is described in detail in [4]. Only a summary is offered here. The particular optimization function we use is

$$\mathrm{Obj}(X) = \sum_i w_i \left(S(t_i) - V_i\right)^2. \quad (2)$$

The objective function uses the square difference between the speech trajectory S and the sequence of ideal targets (visemes) V_i, given at times t_i. The weights w_i are used to give control over how much a target is favored. Essentially, this governs how much a target dominates its neighbors. Note that in the presence of no constraints, w_i will have no impact, and the V_i will be interpolated.

A speech trajectory S will start and end with particular constraints, for example, a neutral state such as silence. These are the boundary constraints, as listed in Table 1, which ensure the articulators in the rest state. If necessary, these constraints can also be used to join trajectories together.

In addition, range constraints can be used to ensure that the trajectory stays within a certain distance of each target,

$$S(t_i) \in [\underline{V}_i, \overline{V}_i], \quad (3)$$

where $\underline{V}$ and $\overline{V}_i$ are, respectively, the lower and upper bounds of the ideal targets V_i.

If (3) and Table 1 are used in (2), the ideal targets V_i will simply be met. A global constraint can be used to dampen the trajectory. We limit the parametric acceleration of a trajectory.

$$\left|S(t)''\right| \le \gamma, \quad \text{where } t \in [t_{\text{start}}, t_{\text{end}}], \quad (4)$$

Table 2: Mexican-Spanish viseme definitions.

Phoneme	Viseme name	Phoneme	Viseme name
Silence	Neutral	i	I
j, g	J	c, k, q	K
b, m, p, v	B_M_P	n, ñ	N
a	A	o,u	O
ch, ll, y, x	CH_Y	r	R
d, s, t, z	D_S_T	l	L

and γ is the maximum allowable magnitude of acceleration across the entire trajectory. As this value tends to zero, the trajectory cannot meet its targets, and thus the w_i in (2) begins to have an effect. The trajectory bends more towards the target, where w_i is high relative to its neighbors. As the global constraint is reduced, the trajectory will eventually reach the limit of at least one range constraint.

The speech trajectory S is represented by a cubic nonuniform B-spline. This gives the necessary C^2 continuity to enable (4) to be applied. The optimization problem is solved using a variant of the sequential quadratic programming (SQP) method (see [6]). The SQP algorithm requires the objective function described in (2). It also requires the derivatives of the objective and the constraints functions: the Hessian of the objective function H_{obj} and the Jacobian of the constraints J_{cstr}. This algorithm follows an iterative process with the steps described in (5). The iterative process finishes when the constraints are met, and there is no further reduction in the optimization function (see Section 5 for discussion of this):

$$\Delta X_{\text{obj}} = -H_{\text{obj}}^{-1} \begin{pmatrix} \dfrac{\partial \text{Obj}}{\partial X_1} \\ \vdots \\ \dfrac{\partial \text{Obj}}{\partial X_n} \end{pmatrix},$$
$$\Delta X_{\text{cstr}} = -J_{\text{cstr}}^{+}(J_{\text{cstr}} \Delta X_{\text{obj}} + C), \tag{5}$$
$$X_{j+1} = X_j + (\Delta X_{\text{obj}} + \Delta X_{\text{cstr}}).$$

3. INPUT DATA FOR THE RANGE CONSTRAINTS

In order to produce specific values for the range constraints described in Section 2, we need to define the visemes that are to be used and measure their visual shapes on real speakers. In English, there is no formal agreement on the number of visemes to use. For example, Massaro defines 17 visemes [9], and both Dodd and Campbell [10], as well as Tekalp and Ostermann [11] use 14 visemes. We chose 15 visemes for Mexican-Spanish, as listed in Table 2.

Many of the 15 visemes we chose are similar to the English visemes, although there are exceptions. The phoneme /v/ is an example, where there is a different mapping between Spanish and English visemes. In English speech, the phoneme maps to the /F/ viseme, whereas in Spanish, the /v/ phoneme corresponds to the /B_M_P/ viseme. There are also letters, like /h/, that do not have a corresponding phoneme in Spanish (they are not pronounced during speech) and thus have no associated viseme. Similarly, there are phonemes in Spanish that do not occur in English, such as /ñ/, although there is an appropriate viseme mapping in this example to the /N/ viseme.

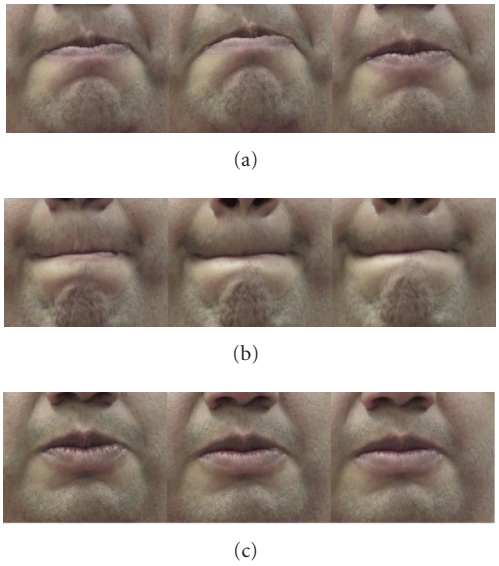
Figure 2: The left two columns show the front and side views of the viseme M. The right two columns show the front and side views of the viseme A. (a) The synthetic face; (b) Person A; (c) Person B; (d) Person C.

To create the range constraints for the Mexican-Spanish visemes listed in Table 2, three native Mexican-Spanish speakers were observed, labeled Person A, Person B, and Person C. Each was asked to make the ideal viseme shapes in Mexican-Spanish, and these were photographed from front and side views. Figure 2 gives examples of the lip shapes for the consonant M (labelled as B_M_P in Table 2) and for the vowel A for each speaker, as well as the modeled synthetic head (which was produced using FaceGen www.facegen.com). Figure 3 shows the variation in the lip shape for the consonant M when Person B pronounces the word "ama" normally, with emphasis and in a mumbling style. This variation is accommodated by defining upper and lower values for the range constraints. Figure 4 illustrates the issue of coarticulation. Person B was recorded three times pronouncing the words "ama," "eme," and "omo," and the frames containing the center of the phoneme "m" were extracted. Figure 4 shows that the shape of the mouth is more rounded in the pronunciation of "omo" because the phoneme m is surrounded by the rounded vowel o.

4. THE SYSTEM

Figure 5 illustrates the complete system for the Mexican-Spanish talking head. The main C++ module is in charge of communication between the rest of the modules. This module first receives text as input, and then gets the corresponding phonetic transcription, audio wave, and timing from a Festival server [12]. The phonetic transcription is used to retrieve the relevant viseme data. Using the information from Festival together with the viseme data, the optimization problem is defined and passed to a MATLAB routine, which contains the SQP implementation. This returns a spline definition and the main C++ module, then generates the rendering of the 3D face in synchronization with the audio wave.

Each viseme is represented by a 3D polygon mesh containing 1504 vertices. Instead of using the optimization process on each vertex, the amount of data is reduced using

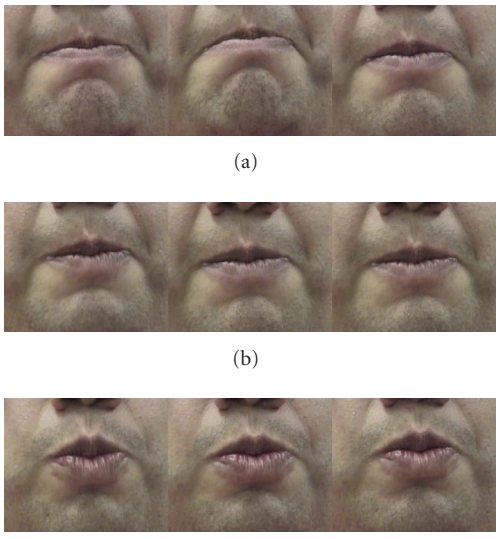


(a)

(b)

(c)

Figure 3: Visual differences in the pronunciation of the phoneme m in the word "ama": (a) normal pronunciation; (b) with emphasis; (c) mumbling. In each case, Person B pronounced the word 3 times to show potential variation.



(a)

(b)

(c)

Figure 4: Contextual differences in the pronunciation of the phonene m: (a) the m of "ama"; (b) the m of "eme"; (c) the m of "omo." In each case, Person B pronounced the word 3 times to show potential.

principal component analysis (PCA). This technique reconstructs a vector V_i that belongs to a randomly sampled vector population V using (6)

$$V = \{v_0, v_1, \ldots, v_s\},$$
$$v_i = u_V + \sum_{j=1}^{s} e_j b_j, \quad 0 \le j \le s, \tag{6}$$

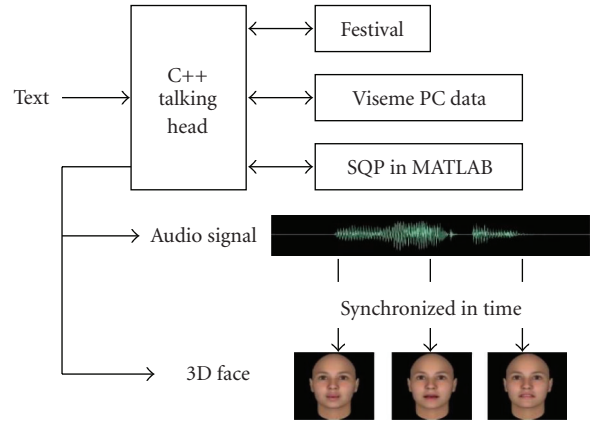


Figure 5: Talking-head system.

where u_V is the mean vector, e_i are the eigenvectors obtained after applying the PCA technique, and b_j are the weight values. With this technique, it is possible to reconstruct, at the cost of minimal error, any of the vectors in the population using a reduced number of eigenvectors e_j and its corresponding weights b_j.

To do the reconstruction, all the vectors share the reduced set of eigenvectors e_j (PCs), but they use different weights b_j for each of those eigenvectors. Thus, each viseme is represented by a vector of weight values.

With this technique, the potential optimization calculations for 1504 vertices are reduced to calculations for a much smaller number of weights. We chose 8 PCs by observing the differences between the original mesh and the reconstructed mesh using different numbers of PCs. Other researchers have used principal components as a parameterization too, although the number used varies from model to model. For example, Edge uses 10 principal components [4], and Kshirsagar et al. have used 7 [13], 8 [14], and 9 [15] components.

It is the PCs that are the parameters (targets) that need to be interpolated in our approach. In the results section, we focus on the PC 1, which relates to the degree that the mouth is open. To determine the range constraints for this PC, the captured visemes were ordered according to the amount of mouth opening. Using this viseme order, the range constraint values were set accordingly using a relative scale. The same range constraint values were set for all other PCs for all visemes. Whilst PC 2 does influence the amount of mouth rounding, we decided to focus on PC 1 to illustrate our approach. Other PCs only give subtle mouth shape differences and are difficult to determine manually. We hope to address this by working on measuring range constraints for static visemes using continuous speaker video. The acceleration constraint is also set for each PC.

5. RESULTS

The Mexican-Spanish talking head was tested with the sentence "hola, cómo estas?". Figure 6 shows the results of the

mouth shape at the time of pronouncing each phoneme in the sentence. Figures 7 and 8 illustrate what is happening for the first PC in producing the results of Figure 6. The pink curves in Figures 7 and 8 show that the global constraint value is set high enough so that all the ideal targets (mouth shapes) are met (visual results in Figure 6(a)). Figure 6(b) and the blue curves in Figures 7 and 8 illustrate what happens when the global constraint is reduced. In Figure 8, the acceleration (blue curve) is restricted by the global acceleration constraint (horizontal blue line). Thus, the blue spline curve in Figure 7 does not meet the ideal targets. Thus, some of the mouth shapes in Figure 6(b) are restricted. The more notable differences are at the second row (phoneme l), at the fifth row (phoneme o), and at the tenth row (phoneme t).

In each of the previous examples, both the global constraint and the range constraint could be satisfied. Making the global constraint smaller could, however, lead to an unstable system, where the two kinds of constraints are "fighting." In an unstable system, it is impossible to find a solution that satisfies both kinds of constraints; and as a result, the system jumps from a solution that satisfies the global constraint to one that satisfies the range constraint in an undetermined way leading to no convergence. To make the system stable under such conditions, there are two options: relax the range constraints or relax the global constraint. The decision on what constraint to relax will depend on what kind of animation is wanted. If we were interested in preserving speaker-dependent animation, we would relax the global constraints as the range constraints encode the boundaries of the manner of articulation of that speaker. If we were interested in producing mumbling effects or producing animation where we were not interested in preserving the speaker's manner of articulation, then the range constraint could be relaxed.

Figure 6(c) and the green curves in Figures 7 and 8 illustrate what happens when the global constraint was reduced further so as to make the system unstable, and the range constraints were relaxed to produce stability again. In Figure 7, the green curve does not satisfy the original range constraints (solid red lines), but does satisfy the relaxed range constraints (dotted red lines). Visual differences can be observed in Figure 6 at the second row (phoneme l), where the mouth is less open in Figure 6(c) than in Figures 6(a) and 6(b). This is also apparent at the fifth row (phoneme o) and at the tenth row (phoneme t).

For Figure 6(d), the speed of speaking was decreased resulting in a doubling of the time taken to say the test sentence. The global constraint was set at the same value as for Figure 6(c), but this time the range constraints were not relaxed. However, the change in speaking speed means that the constraints have time to be satisfied as illustrated in Figures 9 and 10 .

As a final comment, the shape of any facial pose in the animation sequence will be most influenced by its closest visemes. The nature of the constraint-based approach means that the neighborhood of influence includes all visemes, but is at its strongest within a region of 1-2 visemes, either side of the facial pose being considered. This range corresponds to most common coarticulation effects, although contextual effects have been observed up to 7 visemes away [16].

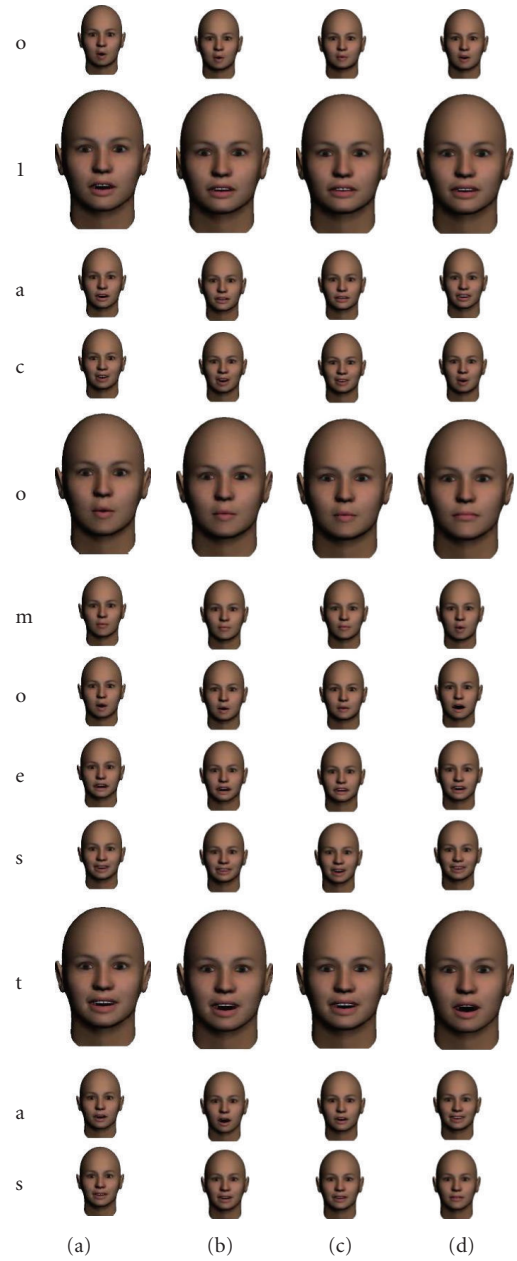


Figure 6: Face positions for the sentence "hola, cómo estas?": (a) targets are met (global constraint 0.03); (b) targets not met (global constraint 0.004); (c) targets not met (global constraint 0.002) and range constraints relaxed; (d) speaking slowly and targets not met (global constraint 0.002).

6. CONCLUSIONS

We have produced a Mexican-Spanish talking head that uses a constraint-based approach to create realistic-looking speech trajectories. The approach accommodates speaker variability and the pronunciation variability of an individual speaker, and produces coarticulation effects. We have demonstrated this variability by altering the global constraint, relaxing the range constraints, and changing the speed of speaking. Currently, PCA is employed to reduce the

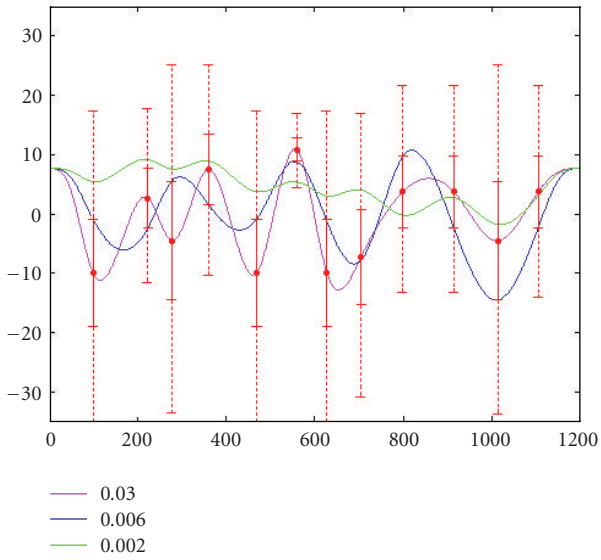


Figure 7: The spline curves for the results shown in Figures 6(a) (pink), 6(b) (blue), and 6(c) (green). The horizontal axis gives time for the speech utterance. The key shows the value of the global acceleration constraint. The red circles are the targets. The solid vertical red bars show the range constraints for Figures 6(a) and 6(b). The dotted bar is the relaxed range constraint for Figure 6(c).

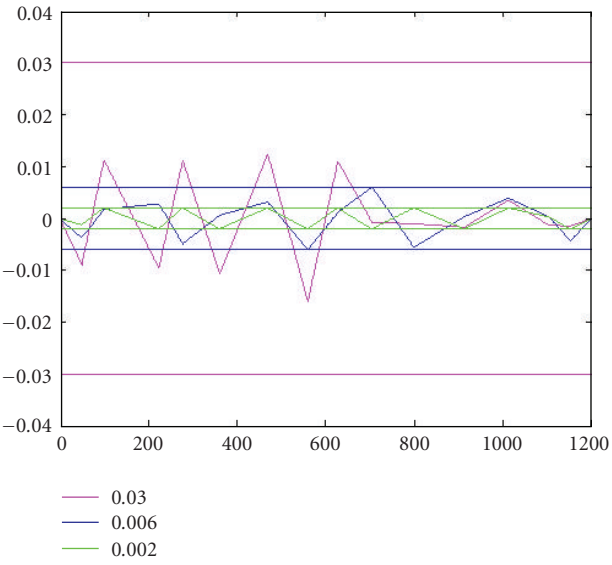


Figure 8: The values of the global acceleration constraints for the results shown in Figures 6(a) (pink), 6(b) (blue), 6(c) (green), and Figure 7. The horizontal axis gives time for the speech utterance. The horizontal lines give the limits of the acceleration constraint in each case.

amount of data used in the optimization approach. However, it is not clear that this produces a suitable set of parameters to control. We are currently considering alternative parameterizations.

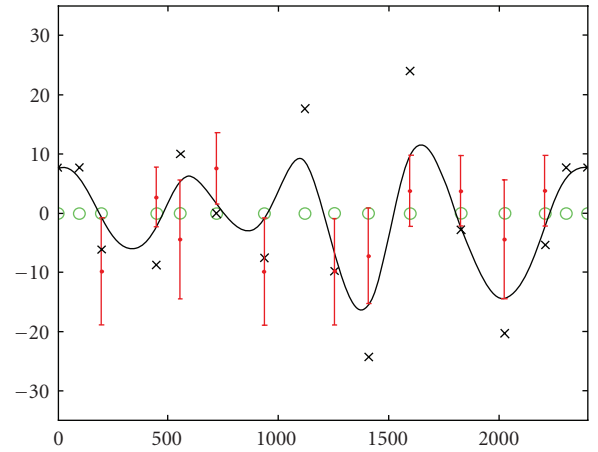


Figure 9: The spline curve for the result shown in Figure 6(d). The global constraint is set to 0.002, and all range constraints are met. The duration of the speech (horizontal axis) is twice as long as Figure 7. The green circles illustrate the knot spacing of the spline, and the x's represent the control points. The solid vertical red bars show the range constraints.

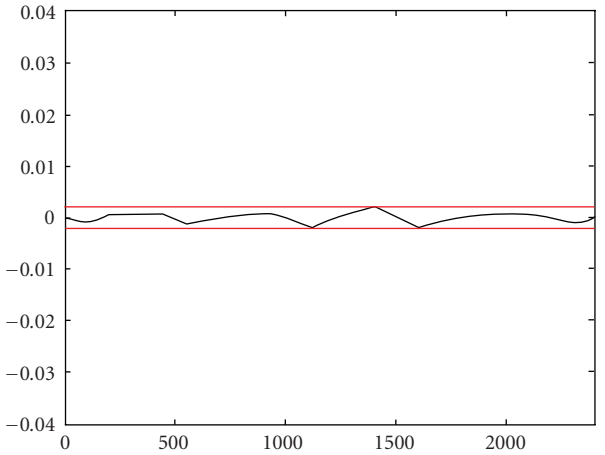


Figure 10: The values of the global acceleration constraint for the result shown in Figure 6(d). The horizontal lines give the limits of the acceleration constraint.

ACKNOWLEDGMENTS

The authors would like to thank Miguel Salas and Jorge Arroyo. They also like to express their thanks to CONACYT.

REFERENCES

[1] F. I. Parke and K. Waters, *Computer Facial Animation*, A K Peters, Wellesley, Mass, USA, 1996.

[2] A. Löfqvist, "Speech as audible gestures," in *Speech Production and Speech Modeling*, W. J. Hardcastle and A. Marchal, Eds., pp. 289–322, Kluwer Academic Press, Dordrecht, The Netherlands, 1990.

[3] M. Cohen and D. Massaro, "Modeling coarticulation in synthetic visual speech," in *Proceedings of the Computer Animation*, pp. 139–156, Geneva, Switzerland, June 1993.

[4] J. Edge, *Techniques for the synthesis of visual speech*, Ph.D. thesis, University of Sheffield, Sheffield, UK, 2005.
[5] J. Edge and S. Maddock, "Constraint-based synthesis of visual speech," in *Proceedings of the 31st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04)*, p. 55, Los Angeles, Calif, USA, August 2004.
[6] A. Witkin and M. Kass, "Spacetime constraints," in *Proceedings of the 15th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88)*, pp. 159–168, Atlanta, Ga, USA, August 1988.
[7] O. M. Lazalde, S. Maddock, and M. Meredith, "A Mexican-Spanish talking head," in *Proceedings of the 3rd International Conference on Games Research and Development (CyberGames '07)*, pp. 17–24, Manchester Metropolitan University, UK, September 2007.
[8] P. E. Gill, W. Murray, and M. Wright, *Practical Optimisation*, Academic Press, Boston, Mass, USA, 1981.
[9] D. W. Massaro, *Perceiving Talking Faces: From Speech Perception to a Behavioral Principle*, The MIT Press, Cambridge, Mass, USA, 1998.
[10] B. Dodd and R. Campbell, Eds., *Hearing by Eye: The Psychology of Lipreading*, Lawrence Erlbaum, London, UK, 1987.
[11] A. M. Tekalp and J. Ostermann, "Face and 2-D mesh animation in MPEG-4," *Signal Processing: Image Communication*, vol. 15, no. 4, pp. 387–421, 2000.
[12] A. Black, P. Taylor, and R. Caley, "The Festival speech synthesis System," 2007, http://www.cstr.ed.ac.uk/projects/festival/.
[13] S. Kshirsagar, T. Molet, and N. Magnenat-Thalmann, "Principal components of expressive speech animation," in *Proceedings of the International Conference on Computer Graphics (CGI '01)*, pp. 38–44, Hong Kong, July 2001.
[14] S. Kshirsagar, S. Garchery, G. Sannier, and N. Magnenat-Thalmann, "Synthetic faces: analysis and applications," *International Journal of Imaging Systems and Technology*, vol. 13, no. 1, pp. 65–73, 2003.
[15] S. Kshirsagar and N. Magnenat-Thalmann, "Visyllable based speech animation," *Computer Graphics Forum*, vol. 22, no. 3, pp. 631–639, 2003.
[16] A. P. Benguerel and H. A. Cowan, "Coarticulation of upper lip protrusion in French," *Phonetica*, vol. 30, no. 1, pp. 41–55, 1974.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 751268, 7 pages
doi:10.1155/2008/751268

Research Article

Activity Classification for Interactive Game Interfaces

John Darby, Baihua Li, and Nick Costen

Department of Computing and Mathematics, The Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M1 5GD, UK

Correspondence should be addressed to John Darby, j.darby@mmu.ac.uk

Received 28 September 2007; Accepted 13 December 2007

Recommended by Kok Wai Wong

We present a technique for modeling and recognising human activity from moving light displays using hidden Markov models. We extract a small number of joint angles at each frame to form a feature vector. Continuous hidden Markov models are then trained with the resulting time series, one for each of a variety of human activity, using the Baum-Welch algorithm. Motion classification is then attempted by evaluation of the forward variable for each model using previously unseen test data. Experimental results based on real-world human motion capture data demonstrate the performance of the algorithm and some degree of robustness to data noise and human motion irregularity. This technique has potential applications in activity classification for gesture-based game interfaces and character animation.

1. INTRODUCTION

The interpretation of human motion is a fundamental task in computer vision. It has received much attention in recent years with wide applications in surveillance, human computer interaction, and the entertainment industry [1]. In vision-based interfaces for video games, such as that in [2] by Decathlete, a player's gestures and activities are used as commands for game control instead of pressing buttons on a keyboard or moving a mouse. In this case, the player's movements, embedded in video images, must be detected, parameterised, and recognised with a sufficient level of accuracy to allow interaction with an intelligent agent.

On the other hand, generating realistic human motion remains an open problem in the game industry. Traditional key-framing methods are extremely labour intensive requiring the manual specification of key poses at specific frames. Physical simulation seems to be more realistic than key-framing, but due to the difficulty of modelling the underlying control mechanism, instabilities, and high computation cost, physics-based animation has not been used with much success. Recently, performance-based animation has received much interest [3]. Among these techniques, marker-based or markerless video-driven animation has shown great potential [4, 5]. Low-level features, such as key-point positions or joint angles, are used to describe full-body movements. MPEG-4, a digital video coding and compression standard primarily used for web-based multimedia applications [6], utilises feature point data as body animation parameters to enhance object-based coding that ultimately facilitates data transmission and storage reduction.

Visual analysis of human motion in video images is a difficult problem in computer vision research. Though progress has been made in the last decade [7–9], marker-free video tracking is still in its infancy in many aspects [1]. Alternatively, marker-based optical motion capture (MoCap) systems are commercially available and have been widely used in the animation industry, such as the Vicon 512 [10]. In this case, motion and structure are presented solely by a small number of moving light displays (MLDs). Despite the complex imaging and vision processing for feature detection from images, we would argue high-level activity recognition information derived from the low-level feature data, such as the MLDs, can be coded more efficiently (than the raw MoCap files) as semantic indexing to enhance human-computer interaction and animation synthesis. Searching and browsing large MoCap file databases is difficult, if not impossible, unless each file is hand labelled with a descriptive string, for example, "run," "walk," and so forth. An interesting question, not only in the context of interaction analysis for computer

games, is how the categorisation and labelling of such data might be automated. If a solution capable of differentiating activities in real-time can be found, then there are also potential applications in interaction representation for games, with user movements controlling avatar animation. The accuracy of the body animation parameters at one extreme, and generic activity classes at the other, with network load of a remote server, for example, the deciding factor.

In this study, we concentrate on a high-level activity recognition task using hidden Markov models (HMMs). Therefore, our algorithm assumes the availability of feature point motion data that might be obtained by various methods and sensors, such as the 3D marker-based optical motion capture data used here. The rest of this paper is organised as follows. Section 2 reviews related work on activity recognition using HMMs. Section 3 describes our choice of feature vector and the use of HMMs for training and classification in the general case. Section 4 provides experimental results on the recognition of human activity. We discuss and conclude our work in Sections 5 and 6.

2. RELATED WORK

Bobick [11] describes three levels of motion understanding problem: *movement*, *activity*, and *action*. For the sake of clarity and cross comparison, we adopt the language of that framework here. The work presented addresses an *activity* recognition problem. We require knowledge of the various *movements* that form the *activities* and the temporal properties of the sequence. We do not attempt to address the questions of context and domain knowledge that allow for the description of *action*.

In the first application of HMMs to human motion recognition, Yamato et al. [12] classified a set of 6 different tennis strokes. They achieve good "familiar person" classification results (better than 90%) but recognition rates drop considerably when the test subject is removed from the training data. This work is also interesting for its use of hidden states with very short duration; they use 36 states for sequences that are between 23 and 70 symbols in length. Wilson and Bobick [13] adopt the HMM in their work on gesture recognition. They are able to recognise simple gestures such as a waving hand. They do not shape the topology of their state transition matrix, for example, by imposing a left-to-right structure on their trained HMM, but leave it potentially ergodic. They argue that although gestures may appear to us as a well defined sequence of conceptual states, they may appear to sensors as a complex mixture of perceptual states. This problem is addressed again by Campbell et al. [14] where the careful selection of features, for example, using velocity rather than position, results in a feature vector that approximates a prototypical trajectory through conceptual states when plotted out in feature space over time. They achieve good results classifying a variety of T'ai Chi moves, but all training and testing data is performed by the same individual, so the generality of the model is not evaluated. Bowden [15] shows that extracting a richer high dimensional feature vector and then performing dimensionality reduction with principal components analysis can help a model to generalise, alleviating the "familiar person" requirement. Brand and Hertzmann [16] introduce stylistic HMMs which specifically address this problem by attempting to recover the "essential structure" of data while disregarding its "accidental properties" in a separation of structure and style.

Brand [17] highlights shortcomings of HMMs for vision research, noting that many activities are not well described by the Markov condition, as they feature multiple interacting processes. He applies a coupled HMM to the classification of T'ai Chi movements, describing the interactions between both hands and shows improved performance over standard HMMs. Galata et al. [18] use variable length Markov models in order to dynamically vary the order of the Markov model. This allows for the consideration of shorter or longer state histories when analysing training data, facilitating the encoding of activity with correlations at different temporal scales.

Outside of the Markovian frameworks discussed in this section, other techniques have been successfully employed for human activity recognition. Section 4 of [1] gives a comprehensive review of the various techniques that have been applied to the action recognition task and a discussion of their relative merits. In particular, both template matching and neural networks have received much attention, for example, [19, 20], respectively. Template matching techniques offer low computational complexity and ease of implementation over state-space approaches such as the HMM. However, they are typically more sensitive to noise and variation in the speed of movements [1]. Neural networks have been shown to be an equally viable approach to human motion classification with near identical results to the HMM [21].

In the context of our own research, we are particularly interested in the HMM for its generative capabilities. The HMM is good for characterizing not only the spatial but also the temporal nature of data. Traversing a trained model gives believable synthetic data. In other work we use this feature of HMMs to provide predictions of a subject's movements in a markerless Bayesian tracking scheme. We believe that although the standard HMM undoubtedly entails consideration of the various shortcomings addressed by the approaches above, and others, it is still a powerful tool and has favourable training requirements versus some of its extensions.

3. METHOD

Human kinematic data used in this work was acquired using a Vicon 512, 3D marker-based optical motion capture system. This provides coordinates of markers attached to feature points on a subject, in the manner of a 3D-MLD system. Feature points are located on the head, torso, shoulders, elbows, wrists, hips, knees, and ankles. The data have been analysed before, with classification achieved by considering the data in the frequency domain [22].

3.1. Feature extraction

In a sequence of frames $m = 1, \ldots, M$ we select a subset of the available feature points. These were the markers on the right shoulder, elbows, wrists, right hip, knees and ankles. Angles

between right radius and right humerus, both radii, right femur and right tibia, and both tibia were then calculated.

For example, the angle between the two radii bones may be calculated from the marker location vectors $\mathbf{m}_{\text{Relb}}$, $\mathbf{m}_{\text{Rwri}}$, $\mathbf{m}_{\text{Lelb}}$, $\mathbf{m}_{\text{Lwri}}$ by defining limb vectors $\mathbf{l}_{\text{Lrad}} = \mathbf{m}_{\text{Lwri}} - \mathbf{m}_{\text{Lelb}}$ and $\mathbf{l}_{\text{Rrad}} = \mathbf{m}_{\text{Rwri}} - \mathbf{m}_{\text{Relb}}$. The relationship

$$|\mathbf{l}_{\text{Lrad}}||\mathbf{l}_{\text{Rrad}}|\cos\theta = \mathbf{l}_{\text{Lrad}} \cdot \mathbf{l}_{\text{Rrad}} \tag{1}$$

is then used to determine the angle θ between limbs. In this way, a feature vector is compiled at each frame (see Figure 1):

$$\mathbf{f}_m = \begin{pmatrix} \theta_{\text{Rrad,Lrad}} \\ \theta_{\text{Rhum,Rrad}} \\ \theta_{\text{Rfem,Rtib}} \\ \theta_{\text{Rtib,Ltib}} \end{pmatrix}, \qquad m = 1, \ldots, M. \tag{2}$$

As limbs are considered relative to one another, the feature vector should remain consistent for a particular pose regardless of the subject's location in the world coordinate system. Although the marker data is unavoidably noisy, this type of feature extraction will provide a tight coupling between conceptual and perceptual states.

3.2. Hidden Markov models

A hidden Markov model can be used to model a time series such as the one derived in the last section. This approach assumes that the underlying system is a Markov process, where the system's state at any timestep m is assumed to depend only on its state at $m - 1$. A standard Markov model is described by a set of states and a set of transition probabilities between these states. The state of the system is allowed to evolve stochastically and is directly observable. This approach may be extended with the introduction of a hidden layer between state and observer. Each state emits an observable symbol from an alphabet common to all states, according to some probability distribution over that alphabet (see Figure 2). This describes a system where both the evolution of the system and the measurement of that evolution are stochastic processes. In our own application HMMs are an appropriate tool as they allow us to handle both the natural variability in a human's performance of a particular activity and also the error of our sensors in estimating their movement.

In order to analyse experimental data using an HMM, we must train HMMs to represent a set of training data and then evaluate the probability that subsequent test data sets were produced by that model. In this way, we may classify a set of N distinct test activities using N HMMs. An HMM λ is specified by parameters S, A_{ij}, A_i, $\text{p}_i(\mathbf{f})$, where

(i) $S = \{s_1, \ldots, s_N\}$ is the set of hidden states;
(ii) the $N \times N$ matrix, A_{ij}, is the probability of a transition from state i to state j;
(iii) A_i is the probability of a sequence starting in state i;
(iv) $p_i(\mathbf{f})$ is the probability of observing feature vector $\mathbf{f}$ while in state i; the emission probability is modelled by a single multivariate Gaussian $p_i(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_i; \boldsymbol{\Sigma}_i) = \exp\{-1/2(\mathbf{f} - \boldsymbol{\mu}_i)^T / \sqrt{(2\pi)^D |\boldsymbol{\Sigma}_i|}\}$ with mean $\boldsymbol{\mu}_i$, covariance $\boldsymbol{\Sigma}_i$, and D the dimensionality of $\mathbf{f}$ (see Figure 3).

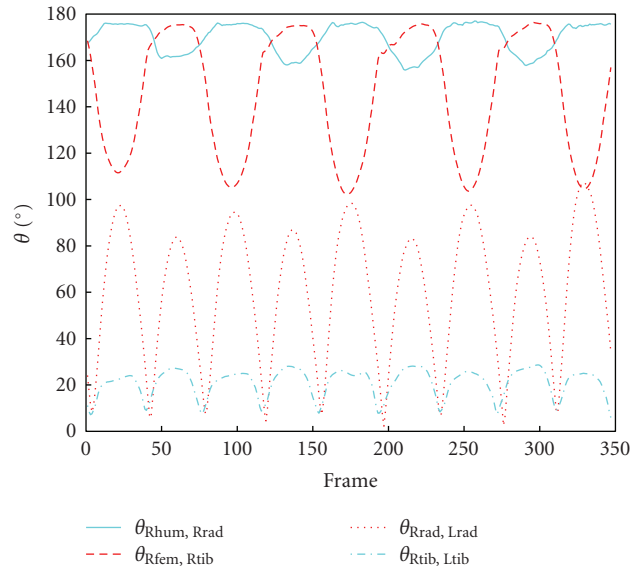


Figure 1: An example of the time series $\mathbf{f}$ for a walking subject.

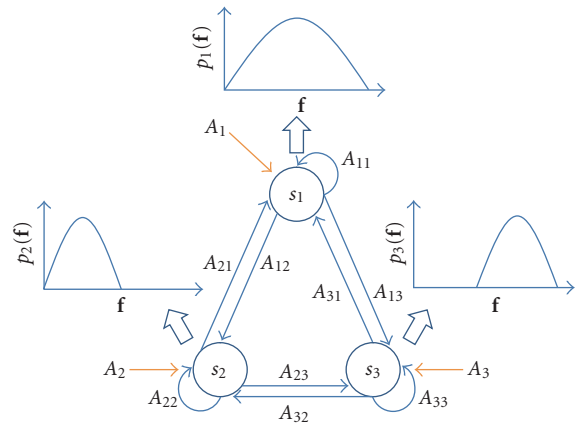


Figure 2: An example of a 3-state HMM with each state emitting a 1D feature vector $\mathbf{f}$.

Sections 3.3 and 3.4 give an overview of the use of continuous HMMs with single multivariate Gaussian observation functions for training and classification.

3.3. Training

Given a feature vector sequence $F = \{\mathbf{f}_1, \ldots, \mathbf{f}_M\}$, we require the set of model parameters that maximise the probability that the data is observed. This problem cannot be solved analytically, but by making estimates of the initial model parameters and applying Baum-Welch reestimation, a form of expectation maximisation, iteration is guaranteed towards a local maximum in $p(F \mid \lambda)$ across the space of models. Although $p(F \mid \lambda)$ may contain a number of critical points, running the algorithm to convergence from a number of

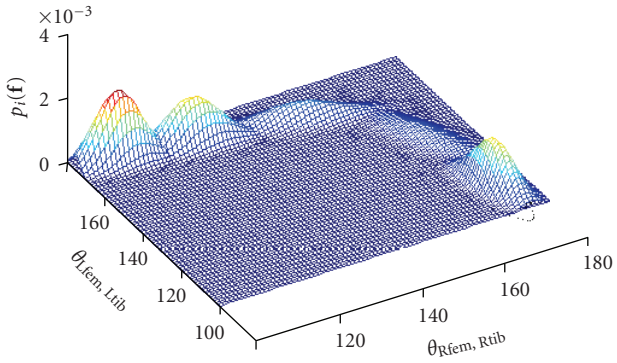


FIGURE 3: $\theta_{\text{Rfem,Rtib}}$ versus $\theta_{\text{Lfem,Ltib}}$ with 5 states.

different estimated initial conditions generally results in a good estimate of the global maximum [23].

The Baum-Welch algorithm requires calculation of the forward and backward variables for the data set F. The forward variable for a state i at time m is the total probability of all paths through the model that emit the training data up to time m, $\{\mathbf{f}_1, \ldots, \mathbf{f}_m\}$ and finish in state i:

$$\alpha_{m,i} = p_i(\mathbf{f}_m) \sum_{j=1}^{N} \alpha_{m-1,j} A_{ji}, \tag{3}$$

where $\alpha_{1,i}$ is calculated using the distribution A_i, that is, $A_i p_i(\mathbf{f}_1)$. Similarly, the backward variable for a state i at time m is the total probability of all paths from state i that emit the rest of the training data $\{\mathbf{f}_{m+1}, \ldots, \mathbf{f}_M\}$:

$$\beta_{m,i} = \sum_{j=1}^{N} \beta_{m+1,j} p_j(\mathbf{f}_{m+1}) A_{ij}, \tag{4}$$

where $\beta_{M,i} = 1$. At any time m, the value $\alpha_{m,i}\beta_{m,i}$ gives the total probability of all paths through the model that produce the data F and pass through state i at time m. Furthermore, $\sum_{i=1}^{N} \alpha_{m,i}\beta_{m,i}$ is constant for all m and gives the probability of the sequence F given λ, or $p(F \mid \lambda)$. We can use these results to calculate the probability that the model was in state s_i when feature vector $\mathbf{f}_m$ was observed, given all the data:

$$\gamma_{m,i} = \frac{\alpha_{m,i}\beta_{m,i}}{\alpha_{m,i}\beta_{m,i}} \tag{5}$$

with which we can estimate the parameters of the Gaussian emission function $p(\mathbf{f})$ associated with each state i:

$$\begin{aligned} \boldsymbol{\mu}_i &= \frac{\sum_{m=1}^{M} \gamma_{m,i} \mathbf{f}_m}{\sum_{m=1}^{M} \gamma_{m,i}}, \\ \Sigma_i &= \frac{\sum_{m=1}^{M} \gamma_{m,i} (\mathbf{f}_m - \boldsymbol{\mu}_i)(\mathbf{f}_m - \boldsymbol{\mu}_i)^T}{\sum_{m=1}^{M} \gamma_{m,i}}, \end{aligned} \tag{6}$$

these are the first two maximisation steps.

In order to reestimate the matrix A_{ij}, we must consider the probability that a transition from state i to state j occurred between timesteps $m-1$ and m:

$$\xi_{m,ij} = p(q_m = s_j, q_{m-1} = s_i \mid F, \lambda) = \frac{\alpha_{m-1,i} A_{ij} p_j(\mathbf{f}_m) \beta_{m+1,j}}{p(F \mid \lambda)}, \tag{7}$$

where q_m is the active hidden state at time m. This is the total probability of all paths through the model which emit $\{\mathbf{f}_1, \ldots, \mathbf{f}_{m-1}\}$ and pass through state i at $m-1$ (given by α_{m-1}), multiplied by the transition-emission pair i transitions to j, j emits $\mathbf{f}_m$, multiplied by the total probability of all paths from state j that emit the remainder of the training data $\{\mathbf{f}_{m+1}, \ldots, \mathbf{f}_M\}$ (given by $\beta_{m,j}$), as a fraction of all paths through the model that emit the data.

By summing over the total number of state transitions, we get the expected number of transitions from i to j:

$$E_{ij} = \sum_{m=2}^{M} \xi_{m,ij}, \tag{8}$$

as the expectation step. The final maximisation step is then

$$A_{ij} = \frac{E_{ij}}{\sum_{j=1}^{N} E_{ij}}. \tag{9}$$

This process can then be iterated, with (6), and (9) providing the new estimate for λ, until some convergence criteria is met. A_i may also be reestimated as $\gamma_{1,i}$ although this is not done in this approach.

3.4. Classification

We can use the definition of the forward variable α in order to calculate the likelihood of a sequence of feature vectors given a particular set of model parameters. For a set of test data $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_M\}$ and model $\lambda = \{S, A_{ij}, A_i, p_i(\mathbf{g})\}$,

$$p(G \mid \lambda) = \sum_{i=1}^{N} \alpha_{M,i}. \tag{10}$$

Therefore, if an HMM is trained for each activity we are interested in recognising, we can evaluate the likelihood that unseen test data was emitted by each of the models and classify data as belonging to the model most likely to have produced it.

4. RESULTS

A set of 6 subjects were recorded performing 6 periodic activities using the Vicon system. These were walking on the spot, running on the spot, one-footed skipping, two-footed skipping, and two types of star jump. Each activity was performed by at least 3 individuals. Each sequence was divided into two halves, each of between 5 to 12 seconds at 60 fps. One half was used for training, the other retained for testing. Although the fact that the motions are periodic is useful as it negates the need to segment the training data, this is not a requirement of the approach. All of the steps described in Sections 4.1 and 4.2 were performed using the HMM Toolbox for Matlab [24].

Table 1: Classification of human activities.

	λ_{Jump1}	λ_{Jump2}	λ_{Run}	λ_{Skip1}	λ_{Skip2}	λ_{Walk}
G_{Jump1}	**17/20**	3/20	0/20	0/20	0/20	0/20
G_{Jump2}	0/20	**20/20**	0/20	0/20	0/20	0/20
G_{Run}	0/15	0/15	**15/15**	0/15	0/15	0/15
G_{Skip1}	0/15	1/15	0/15	**12/15**	2/15	0/15
G_{Skip2}	0/20	0/20	0/20	0/20	**20/20**	0/20
G_{Walk}	0/15	0/15	0/15	0/15	0/15	**15/15**

4.1. Activity training

A feature vector was extracted at each frame as described in Section 3.1. This vector was then extended to contain a finite difference estimate of $\Delta\mathbf{f}_m$ made using the previous timestep, that is, $\Delta\mathbf{f}_m \approx \mathbf{f}_m - \mathbf{f}_{m-1}$. This is helpful in resolving ambiguities such as intersections in the feature vector trajectory, thus reducing the number of states that represent a junction in feature space. It is analogous to a second order HMM, where the previous state as well as current state have an effect on the next transition, thus encapsulating extra "history" in each state of a first order HMM. Each of the activities was represented by 30 states. As in [12] this is a relatively large number considering that each activity has a period of approximately one second. Emitting consecutive conceptual state vectors from the mean point of each state will produce almost identical poses. However, a large number of states helps the initial clustering and provides good results even if it is not intuitively appealing [25].

Initial estimates of the state means and covariance matrices were found by K-means clustering [26]. The transition matrix was initially estimated randomly (with each row of A_{ij} summing to 1) and the prior A_i set with every value equal to $1/N$, where N is the total number of states. A_i was not reestimated in order that test data could begin at any point during the activity unit with no probabilistic penalty. The transition probabilities and state means and covariances were reestimated using no more than 20 iterations of the Baum-Welch update equations of Section 3.3.

4.2. Activity classification

Each subject's test data for each activity was tested separately. Feature vectors were again extracted at each frame to build up a set of observations G. $p(G \mid \lambda)$ was then calculated 5 times for each test sequence, the Baum-Welch algorithm having been allowed to reconverge to a newly estimated set of parameters λ each time. Table 1 summarises the classification results for each batch of activity test data against each trained model. For cross comparison, the forward variable is calculated over the first 2.5 seconds of each test sequence ($M = 150$ in (10)). Classification results are concentrated on the diagonal and no misclassifications are made for 4 of the activities. In the cases of Jump1 and Skip1, all off-diagonal classifications are due to just one test sequence in each batch, with all other sequences being correctly classified. Further discussion of these results is given in Section 5. Using the HMM Toolbox for Matlab, evaluation of $p(G \mid \lambda)$ typically takes between 0.05 to 0.08 seconds, facilitating real-time calculation of $p(G \mid \lambda)$ for the 6 HMMs.

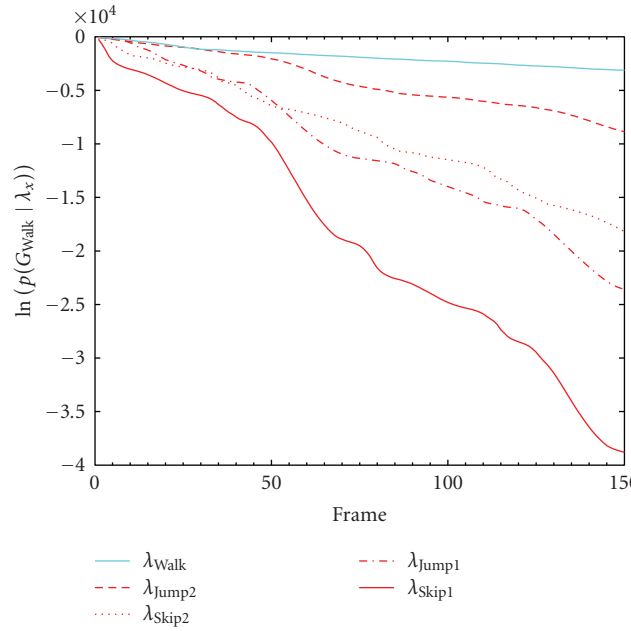


Figure 4: Forward variable for one subject's test walking sequence for all activity models as a function of the number of frames (m).

4.3. Confusion matrices

In the framework outlined in [2], a key aspect of any gesture based interface is its speed in determining a user's activity. Although $p(G \mid \lambda)$ may be calculated in real-time, any approach is limited by the need for sufficient data to stabilise the results of the forward variable evaluations. Determining this data requirement is key to quantifying the level of latency introduced to game play by a gesture based interface. Figure 4 shows the forward variable evaluated using one subject's test walking sequence for each activity model as a function of the number of frames taken as input (m). $p(G_{\text{Walk}} \mid \lambda_{\text{Run}})$ caused arithmetic overflow at $m = 2$ and is not plotted. Walking is not correctly established as the most likely activity until $m = 4$ and jumping temporarily overtakes it for $m = 27, 28, 29$. Walking subsequently remains the most likely interpretation.

Table 2: Classification of two-footed skipping activity versus data segment length.

	λ_{Jump1}	λ_{Jump2}	λ_{Run}	λ_{Skip1}	λ_{Skip2}	λ_{Walk}
$M = 2$	0.0114	0.0193	0.0000	0.0386	**0.9277**	0.0000
$M = 4$	0.0089	0.0114	0.0000	0.0309	**0.9495**	0.0000
$M = 8$	0.0017	0.0017	0.0000	0.0017	**0.9950**	0.0000
$M = 16$	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000
$M = 32$	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000
$M = 64$	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000

In order to determine how quickly reliable classification may take place across the activity cycle, each training sequence was divided into smaller segments for evaluation with the forward variable. Segment lengths of 2, 4, 8, 16, 32, and 64 frames were used and all possible continuous segments of this length tested, with data segments allowed to overlap, thus maximising the number of classification problems considered. The classification results are used to form a confusion matrix for each activity. The confusion matrix for the two-footed skipping activity is shown in Table 2. A correct classification rate of greater than 99% is achieved with a segment size of 8 frames, equivalent to 0.13 seconds of data.

5. DISCUSSION

The learned transition matrices A_{ij} were strongly focused on just a few columns per row. As in [15], no effort was made to number the states meaningfully, for example, in chronological order. However, it would suggest that even though no topology shaping was attempted, Baum-Welch training found a natural left-to-right type structure for the HMM where each state may be self-referential, or may transition to a handful of nearby (in terms of the feature space) states. This supports the claim that the feature vector achieves a tight level of coupling between the conceptual and perceptual states.

Classification between the broad activity types (run, walk, skip, jump) was reliable, but subtle changes in the activity proved harder to classify. For example, the confusion between the two star jumps and one-footed and two-footed skipping seen in the first and fourth row of Table 1 respectively. These activities were only misclassified for one individual's test sequence in each case, and in the case of skipping we believe this to be due to a lack of training data for that subject, causing Baum-Welch training to overfit to the other, longer sequences. However, in the case of the star jumping, the similarity between the two activities, in terms of the feature vector we extract, may mean they are unsuitable for inclusion in a gesture interface as a pair. Included separately, they do not pose a problem.

The compilation of confusion matrices demonstrated that classification was feasible with the consideration of only small amounts of data. The reduction of segment length produced remarkably little spread in the distribution across activity columns of the matrix. Balancing the tradeoff between accuracy and latency in a gesture based interface is an application dependent decision, but confusion matrices compiled in this way should facilitate such development decisions.

Although the models performed well when the individual concerned formed part of the training group, performance worsened significantly when they were removed. Only running on the spot and walking on the spot were consistently recognised. This drop in performance is broadly in line with previous findings, for example, [12]. The resulting models may have failed to recover "underlying structure" due to the high level of variation between training data. Alternatively, they may have suffered from overfitting to what is a small set of training data and an impoverished representation of the activity. In either case, a larger number of people in the training set should improve results.

6. CONCLUSIONS

We have described a technique for classifying human activities with HMMs. In this baseline study, buffered marker data obtained from a MoCap system were successfully used for human activity analysis in real-time. These results demonstrate the proposed method remains a candidate for feature-based on-line recognition tasks in gesture based games.

Although MoCap data is used here, the doubly stochastic nature of the HMM should allow for the use of less invasive, but more noisy, markerless tracking techniques. The HMM may provide a way of interpreting complex user input available from a new generation of computer game input devices, providing a more natural and engaging user experience. This type of high level semantic description of a person's movements could also be incorporated into object based coding schemes such as body animation parameters, as an activity index for decoders.

ACKNOWLEDGMENTS

This research was made possible by an MMU Dalton Research Institute research studentship and EPSRC Grant EP/D054818/1. All MoCap data used in this paper were obtained by an optical motion capture system installed at the Department of Computer Science, University of Wales, UK.

REFERENCES

[1] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, 2003.

[2] W. T. Freeman, D. B. Anderson, P. A. Beardsley, et al., "Computer vision for interactive computer graphics," *IEEE Journal of Computer Graphics and Applications*, vol. 18, no. 3, pp. 42–53, 1998.

[3] A. Menache, *Understanding Motion Capture for Computer Animation and Video Games*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1999.

[4] J. Starck, G. Miller, and A. Hilton, "Video-based character animation," in *Proceedings of the ACM SIGGRAPH Eurographics Symposium on Computer Animation (SCA '05)*, pp. 49–58, Los Angeles, Calif, USA, July 2005.

[5] J. Wilhelms and A. V. Gelder, "Interactive video-based motion capture for character animation," in *Proceedings of the IASTED Conference on Computer Graphics and Imaging (CGIM '02)*, Kauai, Hawaii, USA, August 2002.

[6] Joint Video Team, Information technology—coding of audio-visual objects—part 10: advanced video coding, MPEG-4, ISO/IEC 14496-10, 2005.

[7] A. O. Bălan, L. Sigal, and M. J. Black, "A quantitative evaluation of video-based 3D person tracking," in *Proceedings of the 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS '05)*, vol. 2005, pp. 349–356, Beijing, China, October 2005.

[8] N. Jojic, M. Turk, and T. S. Huang, "Tracking self-occluding articulated objects in dense disparity maps," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 123–130, Kerkyra, Greece, September 1999.

[9] C. Sminchisescu and B. Triggs, "Kinematic jump processes for monocular 3D human tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 1, pp. 69–76, Madison, Wis, USA, June 2003.

[10] Vicon motion systems, http://www.vicon.com/.

[11] A. Bobick, "Movement, activity and action: the role of knowledge in the perception of motion," in *Royal Society Workshop on Knowledge-Based Vision in Man and Machine*, pp. 1257–1265, London, UK, February 1997.

[12] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 379–385, Champaign, Ill, USA, June 1992.

[13] A. D. Wilson and A. F. Bobick, "Learning visual behavior for gesture analysis," in *Proceedings of International Symposium on Computer Vision (ISCV '95)*, pp. 229–234, Coral Gables, Fla, USA, November 1995.

[14] L. W. Campbell, D. A. Becker, A. Azarbayejani, A. F. Bobick, and A. Pentland, "Invariant features for 3-D gesture recognition," in *Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG '96)*, pp. 157–162, Killington, Vt, USA, October 1996.

[15] R. Bowden, "Learning statistical models of human motion," in *Proceedings of the IEEE Workshop on Human Modeling, Analysis and Synthesis (CVPR '00)*, pp. 10–17, Hilton Head Island, SC, USA, July 2000.

[16] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 183–192, New Orleans, La, USA, July 2000.

[17] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 994–999, San Juan, Puerto Rico, USA, June 1996.

[18] A. Galata, N. Johnson, and D. Hogg, "Learning variable-length Markov models of behavior," *International Journal of Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 398–413, 2001.

[19] A. Bobick and J. Davis, "Real-time recognition of activity using temporal templates," in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pp. 39–42, Sarasota, Fla, USA, December 1996.

[20] Y. Guoa, G. Xu, and S. Tsuji, "Understanding human motion patterns," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR '94)*, vol. 2, pp. 325–329, Jerusalem, Israel, October 1994.

[21] I. Boesnach, J. Moldenhauer, C. Burgmer, T. Beth, V. Wank, and K. Bos, "Classification of phases in human motions by neural networks and hidden markov models," in *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems (CCIS '04)*, vol. 2, pp. 976–981, Singapore, December 2004.

[22] B. Li and H. Holstein, "Recognition of human periodic motion-a frequency domain approach," in *Proceedings of the International Conference on Pattern Recognition (ICPR '02)*, vol. 1, pp. 311–314, Quebec, Canada, August 2002.

[23] A. B. Poritz, "Hidden Markov models: a guided tour," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '88)*, pp. 7–13, New York, NY, USA, April 1988.

[24] K. Murphy, Hidden Markov model toolbox for Matlab http://www.cs.ubc.ca/~murphyk/software/HMM/hmm.html.

[25] D. O. Tanguay Jr., "Hidden Markov models for gesture recognition," M.S. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass, USA, 1995.

[26] A. Gersho, "On the structure of vector quantizers," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 157–166, 1982.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 542918, 7 pages
doi:10.1155/2008/542918

Research Article

A Real-Time Facial Expression Recognition System for Online Games

Ce Zhan, Wanqing Li, Philip Ogunbona, and Farzad Safaei

School of Computer Science and Software Engineering, University of Wollongong, NSW 2522, Australia

Correspondence should be addressed to Ce Zhan, cz847@uow.edu.au

Received 31 July 2007; Accepted 31 January 2008

Recommended by Kok Wai Wong

Multiplayer online games (MOGs) have become increasingly popular because of the opportunity they provide for collaboration, communication, and interaction. However, compared with ordinary human communication, MOG still has several limitations, especially in communication using facial expressions. Although detailed facial animation has already been achieved in a number of MOGs, players have to use text commands to control the expressions of avatars. In this paper, we propose an automatic expression recognition system that can be integrated into an MOG to control the facial expressions of avatars. To meet the specific requirements of such a system, a number of algorithms are studied, improved, and extended. In particular, Viola and Jones face-detection method is extended to detect small-scale key facial components; and fixed facial landmarks are used to reduce the computational load with little performance degradation in the recognition accuracy.

1. INTRODUCTION

Multiplayer online games (MOGs) have become popular over the last few years.The collaboration, communication, and interaction ability of MOGs enable players to cooperate or compete with each other on a large scale.Thus, players could experience relationships as real as those in the real world. The "real feeling" makes MOGs attractive to an increasing number of players, despite significant amounts of time and subscription fee required to play. Taking youths in China, for example, according to "Pacific Epoch's 2006 Online Game Report" [1], China had 30.4 million online gamers by the end of 2006.

Despite the advances in interactive realism of MOGs, when compared with real-world human communication, the interfaces are still primitive. For example, in most of the existing MOGs, text-chat is used rather than real-time voice chatting during a conversation, avatars have no activities related to natural body gestures, facial expressions, and so forth.

Among the problems mentioned above, this paper focuses on facial communication in particular. In everyday life, the manifestation of facial expressions is a significant part of our social communication. Our underlying emotions are conveyed by different facial expressions. To feel immersed and socially aware like in the real world, players must have an efficient method of conveying and observing changes in emotional states. Existing MOGs allow players to convey their expressions mainly through text-based commands augmented by facial and body expressions [2].

Although a number of existing MOGs have already achieved detailed animation, text commands do not offer an efficient way to control the avatar's expressions easily and naturally. They are simple and straightforward, but not easy-to-use. First, players have to memorize all the commands. Thus the more sophisticated the facial system is, the harder it is to use. Second, humans convey emotions by expressions in real time. Players cannot type text commands every few seconds to update their current mood. Thirdly, facial communication should happen naturally and effortlessly; typing commands ruins the realism.

The goal of this paper is to automatically recognize the player's facial expressions, so that the recognition results can be used to drive the "facial expression engine" of a multiplayer online game. While many facial recognition systems have been reported, MOGs pose unique requirements on the

system which have not been well addressed. In a summary, a facial expression recognition system for MOGs should meet the following requirements [2].

(i) The recognition has to be performed automatically and in real time.
(ii) The system should consume minimum system resources.
(iii) The system should be robust under different lighting conditions and complex backgrounds.
(iv) The system should be user-independent (e.g., the system should be able to handle users of different genders, ages, and ethnicities).
(v) The input device should be easy to obtain and without any constraints, so only single regular web camera should be used.
(vi) The system should be insensitive to distance of user to camera. (i.e., the system should be able to handle a relatively wide range of face resolutions).
(vii) Players usually have to face the computer screen while playing game. Thus, input of the system should be user's frontal faces with certain degree of tolerance to head rotations.
(viii) Due to entertainment purpose of the system, the recognition accuracy rate need not to be overly conservative.

In this paper, we propose a real-time automatic system that meets the requirements. It recognizes players' facial expressions, so that the recognition results can be used to control avatar's expressions by driving the MOG's "animation engine" instead of text commands. Section 2 provides a brief overview of existing technologies for facial expression recognition. Section 3 describes the proposed system and extension and improvement of several algorithms for an efficient implementation of the system. Section 4 presents the experimental results and Section 5 concludes the paper.

2. OVERVIEW OF FACIAL EXPRESSION RECOGNITION

In computer vision, a facial expression is usually considered as the deformations of facial components and their spatial relations, or changes in the pigmentation of the face. An automatic facial expression recognition system (AFERS) is a computer system that attempts to classify these changes or deformations into abstract classes automatically. A large number of approaches have been proposed since mid 1970s in the computer vision community. Early works have been surveyed by Samal and Iyengar [3] in 1992.Fasel and Luttin [4] and Pantic and Rothkrantz [5] published two comprehensive survey papers which summarized the facial expression recognition methods proposed before 1999. Recently, Tian et al. [6] presented the recent advances (before the year 2004) in facial expression recognition.

Generally, an AFERS consists of three processing stages: face detection, facial feature extraction and representation, and facial expression recognition. The face-detection stage seeks to automatically locate the face region in an input image or image sequences. As the first step of AFERS, its reliability has a major influence on the performance and usability of the entire system. The face detector could detect faces frame by frame or just detect a face in the first frame and then track it in the subsequent images in a sequence.

After the face has been detected, the next step is to extract and represent the information about the facial expression to be recognized. The extraction process forms a high-level description of the expression as a function of the image pixel data. This description commonly referred to as "feature vector" is used for subsequent expression classification. Geometric features which present the shape and locations of facial components and spectral-transform-based features which are gained by applying image filters to face images are often used to represent the information of facial expressions. Irrespective of the kind of feature extraction approach employed, the essential information about the displayed expressions should be preserved. The extracted features should contain high discrimination power and high stability against different expressions.

Facial expression classification is the last stage of AFERS and it is decision procedure.The facial changes can be identified as facial action units (AUs) [7] or six prototypic emotional expressions [8]. Introduced by Ekman and Friesen, each of the six prototypic emotions possesses a distinctive content and can be uniquely characterized by a facial expression. These prototypic emotions are also referred to "basic emotions". They are claimed to be universal across human ethnicities and cultures and comprise happiness, sadness, fear, disgust, surprise, and anger. An AU is one of the 44 atomic elements of visible facial movement or its associated deformation. Ekman and Friesen first use AUs in their facial action coding system (FACS) [9] with the goal of describing all possible perceptible changes that may occur on the face. In applications, a facial expression is represented using a combination of AUs with respect to the locations and intensities of corresponding facial actions.

To attain successful performance, almost all the existing facial expression recognition approaches require some control over the imaging conditions, such as high-resolution faces, good lighting, and uncluttered backgrounds. With these constraints, the existing methods in the literature cannot be directly applied in most real-world applications, which always require operational flexibility. Although deployment of the existing methods in fully unconstrained environments is still in the relatively distant future, integrating and extending these algorithms to develop a facial expression recognition system for a certain application context such as MOG is achievable.

3. PROPOSED SYSTEM

Based on the specific requirements of MOGs, a facial expression recognition system is proposed in this section. The system categorizes each frame of user's facial video sequence into one of the six prototypic emotional expressions.

We hypothesize that recognition of the six prototypic emotional expressions would serve an MOG well in most cases, since players may not have enough time to perceive more subtle facial changes. Figure 1 shows the block diagram of the system, which consists of four components: face

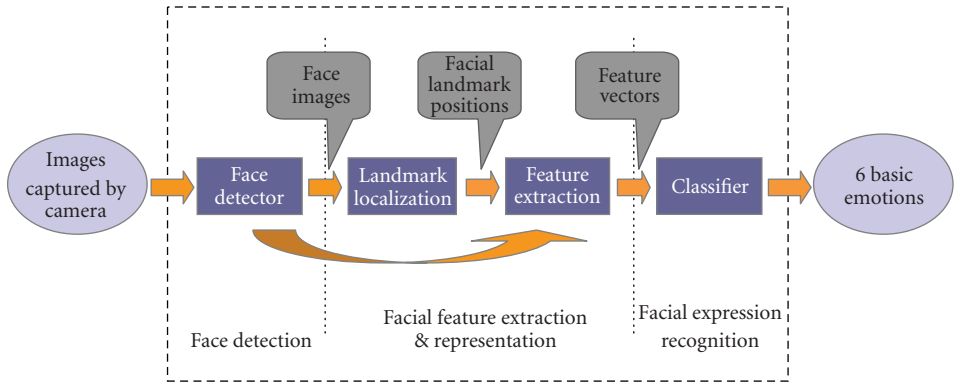


Figure 1: The proposed system for MOGs.

detection, facial landmark localization, feature extraction, and classification of the expressions.

3.1. Face detection

The face region is located in an input image by implementing one of the boosting methods proposed by Viola and Jones [10]. The method achieves real-time detection by using very simple and easily computable Haar-like features; and the good detection rate was obtained by the use of a fundamental boosting algorithm, AdaBoost [11], which selects the most representative features in a large set. As a machine-learning method, most of the time and computational expenditure are consumed during the offline training process. Thus, in the detection process, minimal system resources are needed. The trained face detector scans an image by a subwindow at different scales. Each subwindow is tested by a cascade classifier made of several stage classifiers. If the subwindow is clearly not a face, it will be rejected by one of the first stages in the cascade while more specific classifier will classify it, if it is more difficult to discriminate. For details on the Viola-Jones face-detection method, readers are referred to [10].

3.2. Facial landmark localization

To extract the facial feature automatically, facial landmarks need to be detected without manual efforts. Automatic facial landmark localization is a complex process. To find accurate position of landmarks, most of landmark detection methods involve multiple classification steps and a great number of training samples are required [4, 5]. Although coarse-to-fine localization is widely used to reduce the computational load, the detection process is still too complex and time-consuming for MOGs.

According to the results of the facial landmark location tolerance test conducted in our previous work [2], the facial landmark positions are relatively fixed after the normalization based on three key landmarks: mouth center and eye centers. Thus, it is reasonable to use fixed landmarks on normalized face images rather than performing traditional facial landmark detection; and in this way, only three key facial components are needed to be detected.

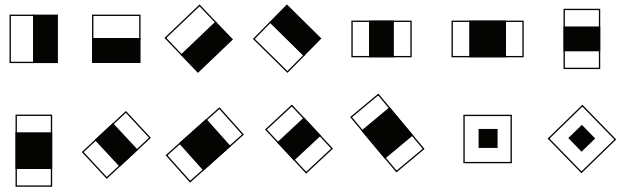


Figure 2: The extended Haar-like feature set.

To take advantage of the computational efficiency of Haar-like features and highly efficient cascade structure used in Viola-Jones Adaboost face-detection method, "AdaBoost" detection principle is still adopted to search the key facial components (the mouth and eyes) within the detected face area. However, low detection rate was observed when the conventional Viola-Jones method was trained with the facial components and employed in the detection process. This is probably due to the lack of structure information of the facial components (compared to the entire face). Especially, the structure of the facial components become less detectable when the detected face is at low resolution. Another possible cause of the low detection rate is the substantial variations of the component shape, especially, mouth, among the different expressions conveyed by the same or different people. This is also true for high-resolution face images. To solve these problems, we improved the "AdaBoost" detection method by employing extended Haar-like features, modified the training criteria, regional scanning, and probabilistic selection of candidate subwindow.

3.2.1. Extended Haar-like feature set

An extended feature set with 14 Haar-like features (Figure 2) based on [12] is used in the facial component detection. Besides the basic upright rectangle features employed in face detection, 45° rotated rectangle features and center-surround features are added into the feature pool. The additional features are more representative for different shapes than the original Haar-feature set, thus they would improve the detection performance.

3.2.2. High hit rate cascade training

In the conventional Viola-Jones method, the cascade classifier is trained based on the desirable hit rate and false-positive rate. Additional stage is added into the cascade classifier if the false positive is higher. However, when the false-positive rate decreases, the hit rate also decreases. In the case of facial components detection, hit rate will dramatically fall for low-resolution face images if the cascade classifier is trained for a target low false-positive rate.

To ensure that low-resolution facial components could be detected, a minimum overall hit rate is set before training. For each stage in the training, the training goal is set to achieve a high hit rate and an acceptable false-positive rate. The number of features used is then increased until the target hit rate and false-positive rate are met for the stage. If the overall hit rate is still greater than the minimum value, another stage is added to the cascade to reduce the overall false-positive rate. In this way, the trained detectors will detect the facial components at a guaranteed hit rate though some false positives will occur, which can be reduced or removed by the scanning scheme introduced below.

3.2.3. Regional scanning with a fixed classifier

Rather than rescaling the classifier as proposed by Viola and Jones, to achieve multiscale searching, input face images are resized to a range of predicted sizes and a fixed classifier is used for facial component detection. Due to the structure of face, we predict the face size according to the size of facial component used for training. In this way, the computation of the whole image pyramid is avoided. If the facial component size is larger than the training size, fewer false positives would be produced due to down sampling; when the component is smaller than the training sample, the input image is scaled up to match the training size.

In addition, prior knowledge of the face structure is used to partition the region of scanning. The top region of the face image is used for eye detection, and the mouth is searched in the lower region of the face. The regional scanning not only reduces the false positives, but also lowers the computation.

3.2.4. Candidate subwindow selection

To select the true subwindow which contains the facial component, it is assumed that the central position of the facial components among different persons follows a normal distribution. Thus, the probability that a candidate component at $\mathbf{k} = [x \quad y]^T$ is the true position can be calculated as

$$P(\mathbf{k}) = \frac{1}{(2\pi)|s\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{k} - s\mathbf{m})^T s\boldsymbol{\Sigma}^{-1}(\mathbf{k} - s\mathbf{m})\right), \tag{1}$$

where the mean vector $\mathbf{m}$ and the covariance matrix $\boldsymbol{\Sigma}$ are estimated from normalized face image dataset. The scale coefficient "s" can be computed as $s = w_d/w_n$; w_d is the width of detected face; and w_n is the width of normalized training faces. The candidate with maximum probability is selected as the true component.

Figure 3: The landmark localization process: (from left to right) detection of face and facial components, normalised face, and fixed set of facial landmarks on the normalised face.

3.2.5. Specialized classifiers

Two cascade classifiers are trained for mouth. One is for detecting closed mouths, and the other is for open mouths. During scanning, if the closed mouth detector failed to find a mouth, the open mouth detector is triggered. In addition, the left and right eye classifiers are trained separately.

After the area of key facial components, mouth and eyes, have been detected, face images are normalized based on the centers of the components; and finally, mean coordinates of facial landmarks obtained from the "location tolerance test" are used as landmarks. Figure 3 shows the landmark localization process.

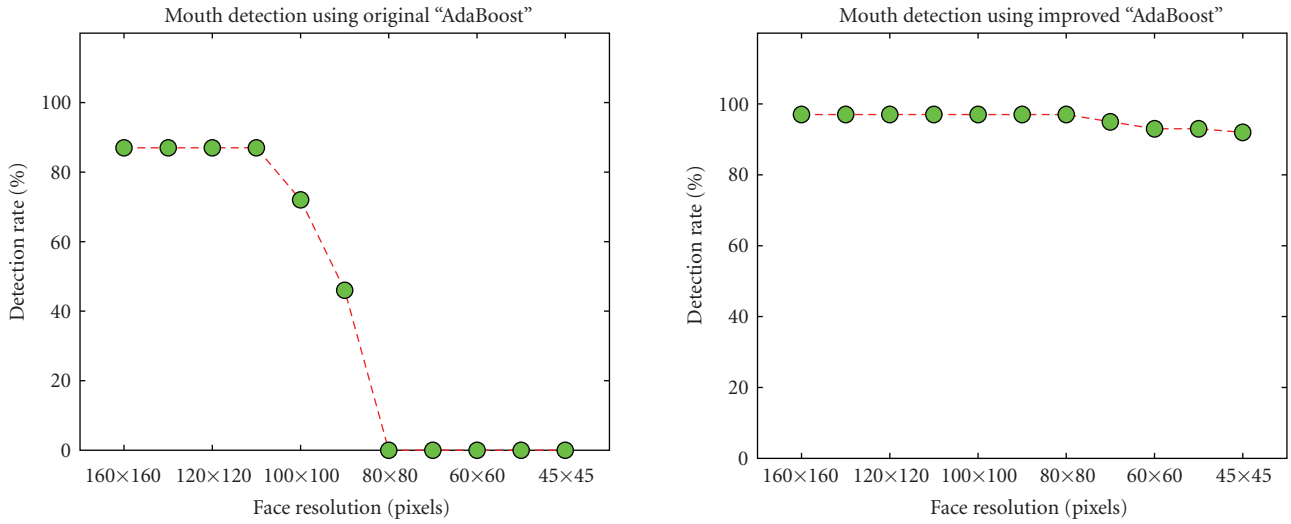
3.3. Feature extraction

As stated previously, the extracted features should possess high discriminative power and high stability against different expressions. Among a number of feature extraction algorithms proposed in the literature, research has demonstrated that Gabor filters are more discriminative for facial expressions and robust against various types of noise than other methods [4]. However, applying Gabor filters to the whole face area is too costly for MOGs. In the proposed system, Gabor filters with different frequencies and orientations are applied only to a set of facial landmark positions. Thus, not only the real-time requirement can be met due to the reduced amount of data to be processed, but also the limited localization in space and frequency yields a certain amount of robustness against translation, distortion, rotation, and scaling of the images. At the same time, face cropping or alignment is not necessary in the whole recognition process since feature extraction is conducted at specific locations.

A 2D Gabor function is a plane wave with wave vector $\mathbf{k}$, restricted by a Gaussian envelope function with relative width σ:

$$\Psi(\mathbf{k}, \mathbf{x}) = \frac{\mathbf{k}^2}{\sigma^2} \exp\left(-\frac{\mathbf{k}^2\mathbf{x}^2}{2\sigma^2}\right)\left[\exp(i\,\mathbf{k}\cdot\mathbf{x}) - \exp\left(-\frac{\sigma^2}{2}\right)\right]. \tag{2}$$

In our implementation, we set $\sigma = \pi$ [13]. A set of Gabor kernels, which comprises 3 spatial frequencies ($\mathbf{k} = \pi/4$, $\pi/8$, $\pi/16$) and 6 different orientations ($\pi/6$, $2\pi/6$, $3\pi/6$, $4\pi/6$, $5\pi/6$, π) [14], is employed. The parameters of Gabor kernels are chosen based on a large number of experiments, so that the extracted feature vectors only contain the most important components with high discriminative power. Each image is convolved with both even and odd Gabor kernels



(a) Original "AdaBoost" (b) Improved "AdaBoost"

Figure 4: Mouth detection result. Both detectors are trained using same dataset.

at facial landmarks (as shown in Figure 3). Thus, 18 complex Gabor wavelet coefficients are obtained at each landmark. Since only magnitudes of these coefficients are used, each face image is represented by a vector of 360 ($3 \times 6 \times 20$) when 20 landmarks are used.

3.4. Classification

A wide range of classifiers in pattern recognition literature have been applied to expression classification. We evaluated a number of classification methods in [2]. In this paper, support vector machines (SVMs) [15] are employed.

SVMs belong to the class of kernel-based supervised learning machines and have been successfully employed in general-purpose pattern-recognition tasks. Based on statistical learning theory, SVMs find the biggest margin to separate different classes. The kernel functions employed in SVMs are used to efficiently map input data which may not be linearly separable to a high-dimensional feature space where linear methods can then be applied. Since there are often only subtle differences between different expressions posed by different people, for example,"anger" and "disgust" are very similar. The high discrimination ability of SVMs plays a major role in designing classifiers that can distinguish such expressions. SVMs also demonstrate relatively good performance when only a modest amount of training data is available, and this also makes SVMs suitable for the system under consideration. Furthermore, only inner products are involved in SVMs computation; the learning and prediction processes are much faster than some traditional classifiers such as a multilayer neural network.

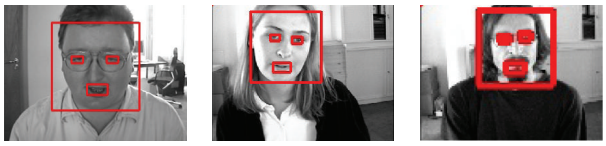
In the implementation, classifiers are trained to identify Gabor coefficient vectors obtained from feature extraction process into one of the six basic emotional expressions or a neutral expression. Since SVMs are binary classifiers and there are 7 categories to distinguish, 21 SVMs are trained to discriminate all pairs of expressions. A multiclass classifier is obtained by combining the SVM outputs through a voting principle. For example, if one SVM makes the decision that the input is "Happiness" and not "Sadness," then happiness gets +1 and sadness gets −1. After all SVMs have made their decisions, votes for each category are summed together, and the expression with the highest score is considered to be the final decision.

4. EXPERIMENTAL RESULTS

4.1. Facial component detection

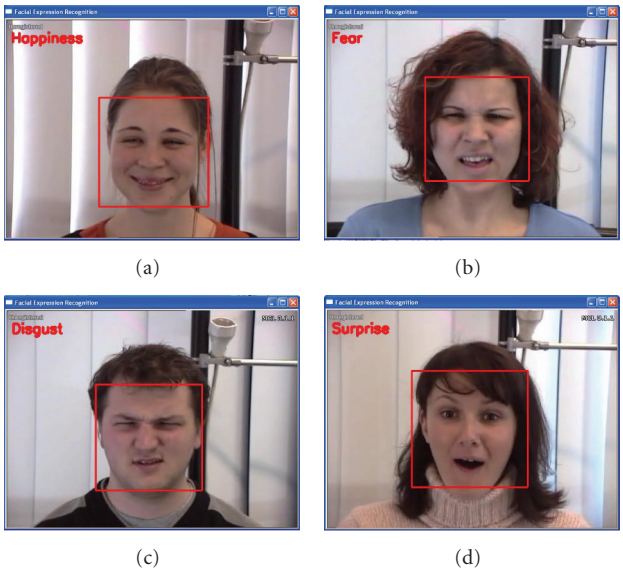
As introduced in Section 3.2, 4 cascade classifiers were trained to detect the key facial components, one for left eyes, one for right eyes, and two for mouths. Positive training samples of eyes and mouths and negative samples (nonfacial components) were cropped from AR database [16] and AT&T database [17]. To accommodate low-resolution facial components, the training samples were rescaled to small sizes: 10×6 for eyes and 16×8 for mouth. For each detector, about 1200 positive samples and 5000 negative samples were used for training.

The trained detectors were tested on BioID database [18]. To evaluate the performance on low-resolution input, the test images were down sampled to different resolutions to simulate low-resolution faces which are not included in the database. To show the improvement compared with the original detection method proposed by Viola and Jones, mouth detection results at different face resolutions are presented in Figure 4. The average left eye, right eye, and mouth detection rate for different face resolutions is 95.7%, 97.2%, and 95.6%, respectively. A few detection examples are shown in Figure 5.



(a) 200×200 (b) 100×100 (c) 50×50

Figure 5: Facial component detection results for different resolution faces from BioID database.



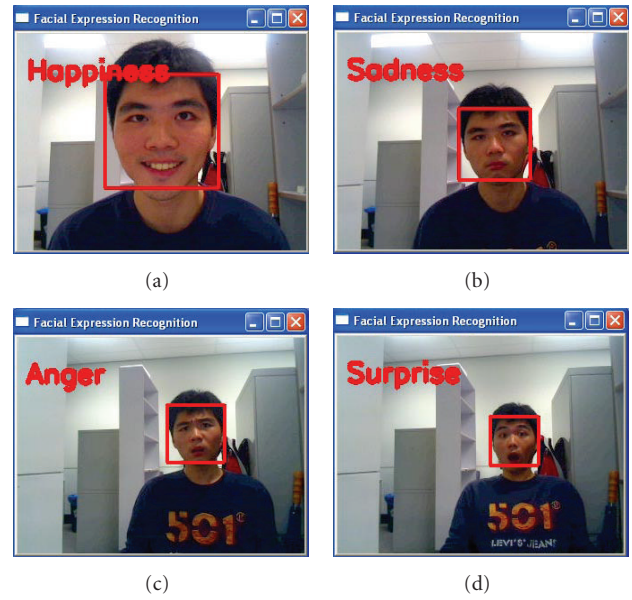
(a) (b)

(c) (d)

Figure 6: Recognition samples from FG-NET.

4.2. Expression recognition

FG-NET database [19] was used in the experiment. The database contains 399 video sequences of 6 prototypic emotional expressions and a neutral expression from 18 individuals. For each expression of each person, at least 3 sequences are provided. In the experiment, one sequence of each expression is left out for test, and the rest are used as the training samples. The recognition result is presented in Table 1 and some samples are shown in Figure 6. The results show that "Happiness," "Surprise," and "Neutral" are detected with relative high accuracy while other more subtle expressions were a little bit harder to recognize, especially for "Sadness". During testing, we found that "Sadness," "Anger," "Fear," and "Disgust" are confused with each other frequently, sometimes even human beings are not able to discriminate them, however, they are seldom confused with other expressions. Thus, if these four expressions are treated as one, together with "Happiness," "Surprise," and "Neutral," we can estimate user's emotional state more accurately on a higher level. Naming the new expression as unhappy, classification result for 4 expressions are presented in Table 2. In this way, the system is able to tell with an 85.5% accuracy if the user is in good mood, bad mood, or just surprised. We also tested the system in practical conditions, some samples are shown in Figure 7. The results show that the system is relatively robust against complex background and lighting conditions, furthermore, it works on the images taken from a practical range of distances from user to camera.



(a) (b)

(c) (d)

Figure 7: Recognition samples for real-time test.

Table 1: Recognition results for 7 expressions classification.

Expression	Recognition rate
Happiness	85.2%
Sadness	78.9%
Fear	80.7%
Disgust	81.6%
Surprise	86.3%
Anger	83.3%
Neutral	84.9%

Table 2: Recognition results for 4 expressions classification.

Expression	Recognition rate
Happy	85.2%
Unhappy	85.6%
Surprise	86.3%
Neutral	84.9%

5. MOG IMPLEMENTATION ISSUES

In this section, we indicate the manner in which the proposed system can be incorporated in an MOG. A typical MOG is a complex distributed system connecting thousands of users. Two main types of network architecture are employed, namely, client-server and peer-to-peer [20]. We refrain from any comparative discussion about the two types of architecture since this paper is not about such considerations.

The system presented in this paper is implemented on the client side as it constitutes a user interface device enhancement. The system outputs a classification of the current emotion of the player and this is transmitted to the server. It is possible that an XML-based description of the emotions is employed. The game logic server running of the centralized server would incorporate a module that can parse the XML message and send the appropriate message to the game world module which in turn issues the necessary message that allows the correct view of the avatar to be generated. Thus, the facial expression recognition system allows a rendering of the appropriate avatar with the required emotion on clients' world views.

6. CONCLUSIONS

In this paper, we presented an automatic facial expression recognition system for MOGs. Several algorithms are improved and extended to meet the specific requirements. Despite recent advances in computer vision techniques for face detection, facial landmarks localization, and feature extraction, building a facial expression recognition system for real-life applications still remains challenging.

REFERENCES

[1] http://www.pacificepoch.com/.

[2] C. Zhan, W. Li, F. Safaei, and P. Ogunbona, "Facial expression recognition for multiplayer online games," in *Proceedings of the 3rd Australasian Conference on Interactive Entertainment*, vol. 207, pp. 52–58, Perth, Australia, December 2006.

[3] A. Samal and P. A. Iyengar, "Automatic recognition and analysis of human faces and facial expressions: a survey," *Pattern Recognition*, vol. 25, no. 1, pp. 65–77, 1992.

[4] B. Fasel and J. Luettin, "Automatic facial expression analysis: a survey," *Pattern Recognition*, vol. 36, no. 1, pp. 259–275, 2003.

[5] M. Pantic and L. J. M. Rothkrantz, "Automatic analysis of facial expressions: the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1424–1445, 2000.

[6] Y.-L. Tian, T. Kanade, and J. F. Cohn, *Hand Book of Face Recognition*, Springer, New York, NY, USA, 2004.

[7] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, "Classifying facial actions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 974–989, 1999.

[8] P. Ekman, *Emotion in the Human Face*, Cambridge University Press, New York, NY, USA, 1982.

[9] P. Ekman and W. Friesen, *Facial Action Coding System (FACS):Manual*, Consulting Psychologists Press, Palo Alto, Calif, USA, 1978.

[10] P. Viola and M. J. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[11] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," in *Proceedings of the 2nd European Conference on Computational Learning Theory (EuroCOLT '95)*, pp. 23–37, Barcelona, Spain, March 1995.

[12] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings of the International Conference on Image Processing (ICIP '02)*, vol. 1, pp. 900–903, Rochester, NY, USA, September 2002.

[13] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expression with gabor wavelets," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 200–205, Nara, Japan, April 1998.

[14] Z. Zhang, M. Lyons, M. Schuster, and S. Akamatsu, "Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 454–459, Nara, Japan, April 1998.

[15] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[16] http://cobweb.ecn.purdue.edu/aleix/aleix_face_DB.html.

[17] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[18] http://www.bioid.com/.

[19] http://www.mmk.ei.tum.de/ waf/fgnet/feedtum.html.

[20] S. Bogojevic and M. Kazemzadeh, "The architecture of massive multiplayer online games," M.S. thesis, Lund Institute of Technology, Lund University, Lund, Sweden, 2003.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 243107, 9 pages
doi:10.1155/2008/243107

Research Article

Perception-Based Filtering for MMOGs

Souad El Merhebi, Jean-Christophe Hoelt, Patrice Torguet, and Jean-Pierre Jessel

VORTEX Group, Institut de Recherche en Informatique de Toulouse, Paul Sabatier University, 31062 Toulouse, France

Correspondence should be addressed to Souad El Merhebi, merhebi@irit.fr

Received 30 September 2007; Revised 13 January 2008; Accepted 8 April 2008

Recommended by Kok Wai Wong

Online games have exploded in the last few years. These games face several problems linked to scalability and interactivity. In fact, online games should provide a quick feedback of users' interactions as well as a coherent view of the shared world. However, the search for enhanced scalability dramatically increases message exchange. Such an increase consumes processing power and bandwidth, and thus limits interactivity, consistency, and scalability. To reduce the rate of message exchange, distributed virtual environment systems use filtering techniques such as interest management that filters messages according to users' interests in the world. These interests are influenced by perceptual facts which we study in this paper in order to build upon them a perception-based filtering technique. This technique satisfies users' needs by precisely providing an exact filtering which is more efficient than other techniques.

1. INTRODUCTION

Online game players interact within a shared virtual world through their avatars. Players' interactions change the state of their avatars and may affect the state of the world. To insure coherency, all players should observe the same state of the shared world at the same moment. Therefore, the activity of a player must be communicated to the other players. This must be done as quickly as possible to provide an interactive game in which players have a quick feedback of their interactions.

On the other hand, online games, especially massively multiplayer online games (MMOGs), need to be more and more scalable. However, increasing the number of participants tends to dramatically increase the number of update messages. A higher amount of message exchange requires higher bandwidth and more processing power, which slows down the system and makes it less interactive "lagging" [1]. The system should therefore be adapted to these limited resources. This can be done by minimizing exchanged messages and the number of their recipients.

Distributed virtual environments (DVEs) systems take advantage of the fact that participants neither perceive nor are interested in the whole virtual world. This is due to many reasons: world's design, distances that separate avatars, limitation of avatars' awareness, and so forth. Sometimes, developers themselves create these limitations to enhance the scalability of the system (shards, fog, etc.). Limitations of awareness define the interests of players within the world. This provides an interesting basis to design message filtering whose concept is to deliver messages only to concerned participants, this is called interest management.

The perception of an avatar within its world expresses or contributes to expressing its interest. Perception-based filtering [2] defines the perceptual interests of avatars in an exact way. This paper completes the study of perception-based filtering by including a more general perception model and providing deeper mathematical analysis backed up with the results of an experimental evaluation.

In Section 2, we present a brief overview of related work. Section 3 describes the relation between awareness and perception. Section 4 describes perception-based filtering. In Section 5, we present the translation of perception rules into existing interest management techniques. Section 6 presents the experimental and analytical evaluation of perception-based filtering. Finally, Section 7 presents conclusions and outlines future work.

2. RELATED WORK

Interest management techniques have been implemented within military DVE systems as well as in games. They use the division of space or the definition of individual interests of avatars, or a combination of the two approaches.

NPSNET [3] divides the environment into hexagonal cells and specifies for each entity an area of interest (AOI), which is a circle centered on the entity. An entity is only aware of entities within the hexagons that overlap its AOI.

MASSIVE implements the spatial model of interaction [4] which assigns an aura to each object for each medium (graphics, audio, etc.). Spatial model's filtering is achieved in two stages. The first detects auras' collision of couples of objects. This entails a possibility of interaction and leads to the second stages which consists, in the negotiations, of levels of awareness between users. The level of awareness of entity A toward entity B is a function of A's focus and B's nimbus, the focus being the observer's allocation of attention and the nimbus the observed's manifestation or observability. The negotiations of awareness levels decide the *existence* of awareness on each side. If there is awareness, its level can influence the attribution of resources (e.g., bandwidth, audio quality, etc.). Whenever a change occurs, the couple has to negotiate again to know whether a change in awareness took place. For performance and development cost reasons, this model is often used in a simplified way which only uses auras to determine awareness of avatars, even if this entails the exclusive establishment of symmetrical relations.

Effect management [5] is a variation of the spatial model of interaction which preserves the attractive feature of asymmetric awareness, while offering the possibility of not defining foci and nimbi functions if there are no resource accommodations. Effect management determines the existence of awareness on the two sides distinctly in the first stage. The second stage (i.e., awareness level computation) will only be dedicated to the determination of the *nonnull* awareness level through foci and nimbi negotiations if needed. To achieve that, the first stage associates with each entity a viewing zone which reflects the ability to perceive and an effect zone which reflects the ability to be perceived instead of combining the two abilities within the aura. When the viewing zone of an entity A overlaps the effect zone of an entity B, A becomes aware of B.

HLA [6, 7] uses interest management through a runtime infrastructure (RTI) service: the data distribution management (DDM). This service first allows the expression of entities' interest and availability by defining subscription and publication zones. Then, it computes intersections between zones and deliver updates according to those computations (i.e., an update is only sent to the entities whose subscription zones intersect with its publication zone). DDM uses a multidimensional coordinate system in which entities define their interests and availability. This system can use a fixed or hierarchical grid.

Other DVE systems and online games use division of space without the individual definition of entities' interests into zones. Some of them also associate to each division or share a server in order to enhance scalability. Such systems are SPLINE [8], RING [9], Ultima Online, Neverwinter Nights, Silkroad Online, World of Warcraft, and so forth.

Our study focuses on the definition of individual interests of avatars because this approach provides a more exact filtering than division of space. As we have seen, many interest management techniques define their users' interests in individual zones because they are simple, natural and not expensive. However, these systems define the extents of their zones in an empirical way. This leads to imprecise filtering which makes avatars receive more or fewer messages than they actually need. In fact, if an avatar does not receive all the messages that interest it, the player's experience will be less realistic and interactive. Otherwise, if avatars receive messages that do not interest them, the system and the network will be overloaded with unnecessary messages. Thus, in both cases the empirical definition of zone extents will harm the simulation. For these reasons, it is very important to have a mechanism that determines the interests of users in a precise way. Our previous study of perception-based filtering [2] defined the limits of avatars' perception and presented a translation of these limits into zones in other interest management techniques. In this paper, we will complete this study by presenting a more general perception model and by providing a comparative analysis backed up with experimental results between the perception-based filtering and the other interest management techniques.

3. RELATION BETWEEN AWARENESS AND PERCEPTION

The perceptibility of an avatar (its ability to be perceived, its manifestation) depends on its importance in the world. This importance may be related to the avatar's situation (an aircraft is easily perceived in a clear sky) or its function (a speaker in a stadium is seen and heard by all the audience). Most importantly, an avatar is noticed due to its physical manifestation: its size.

On the other hand, not all avatars have the same capacity of awareness because of their structure, function or behavior. An avatar can simulate a short-sighted person, another represents a person carrying binoculars. We can also have a well-positioned entity or a fast entity. These entities have different degrees of perception.

Therefore, an avatar's awareness of another depends on the perception of the observer, the perceptibility of the observed and the distance that separates the couple. This awareness may only depend on these factors in case of the absence of others. This relationship was expressed in some interest management techniques such as area of interest management, spatial model of interaction, and effect management through the use of zones.

4. PERCEPTION-BASED FILTERING

Computer graphics' developers use many algorithms to perform visibility tests. In the following, we will study these algorithms to see if they may specify convenient message filtering parameters.

Computer graphics determines that an object is not visible due to the following widely used properties.

(i) P1 is outside of camera's view volume.

(ii) P2 is hidden by other objects.

(iii) P3 covers less than one pixel on the screen.

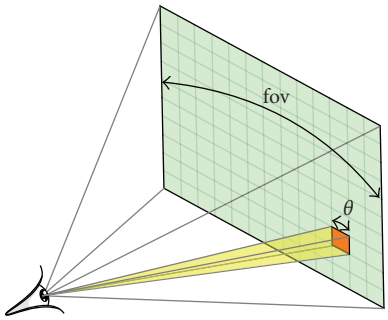


Figure 1: Angle of view of a virtual camera.

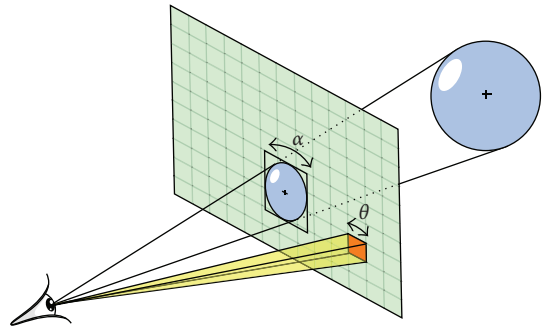


Figure 2: Angular extents of a pixel and a sphere.

P1 cannot define an efficient filtering technique for DVEs because when a player turns his head, he may quickly visualize a lot of new avatars. Therefore, P1 does not define stable filtering parameters that prevent an avatar from frequent and thus expensive changes of interests.

P2 may be interesting because it can eliminate an important amount of message exchange. Many techniques, based on this property, have been used in computer graphics. These techniques can be used for message filtering for DVEs. For example, it is possible to use occlusion queries on recent GPUs [10] or precomputed potentially visible sets [11] (as used in Quake(tm) [12] and RING [9], e.g.). However, the implementation of these techniques is difficult and they require complex computations when they are done dynamically (especially on a client/server architecture because the server performs filtering for many entities during each simulation frame).

P3 offers a simpler method: knowing player A's screen resolution and the 3D-engine camera parameters, we can easily determine the projection of avatar B's bounding sphere on the screen. If the size of the projected sphere is less than a certain threshold (which may be set to the size of one pixel, e.g.), A will not see B. This will be the basis of our perception-based filtering.

4.1. Maximal distance of perception

Let us take the standard approach of a virtual camera using a projection plane as it is defined in OpenGL and Direct3D. The camera is defined by its angle of view (or field of view—fov), which is the angular extent of the virtual scene that is rendered (see Figure 1). This defines a viewing volume (a pyramid), which is then divided into smaller volumes, one for each pixel. Given this, we can determine the angle of view covered by a pixel for a specific player from the virtual camera's field of view (fov) and the number of pixels covered by the user's viewport (SSize). The whole screen covers 100% of the field of view, therefore a single pixel covers approximately an average angle θ (The camera model induces a distortion for the pixels that are far from the center of the screen. We will consider this distortion negligible in our study.):

$$\theta = \frac{\text{fov}}{\text{SSize}}. \tag{1}$$

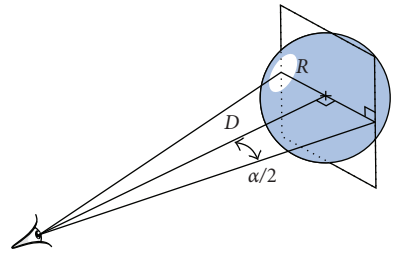


Figure 3: Angular extent of a 3D object.

On the other hand, we can compute the angular extent α of a 3D object projection on the screen (Figure 2). To compute this extent, we can see in Figure 3 that the relationship between the angle α (view angle covered by the object), the "opposite" side R (the bounding sphere radius), and the "adjacent" side D (distance between object and camera) is

$$\tan\left(\frac{\alpha}{2}\right) = \frac{R}{D}. \tag{2}$$

By using our filtering criteria, P3 allows an object to be seen when the size of its projection is superior to one or several pixels. The minimal number of perceived pixels depends on the viewing capacity of the user. A user with an enhanced capacity of view can see an object if its projection covers one pixel, while a short-sighted user will need a projection superior to several pixels to be able to see a projected object.

When the size of the projected sphere is bigger than a pixel this is exactly the same as having the angular extent "α" bigger than the angle covered by one pixel "θ" (Figure 2). For a short-sighted user, this maximal angle will be multiplied by the minimal number of pixels seen by the user "n". Knowing that α and $n\cdot\theta$ are always smaller than π, we have

$$\begin{aligned} \alpha \geq n\cdot\theta &\Longleftrightarrow \tan\left(\frac{\alpha}{2}\right) \geq \tan\left(n\cdot\frac{\theta}{2}\right) \\ &\Longleftrightarrow \frac{R}{D} \geq \tan\left(n\cdot\frac{\theta}{2}\right) \\ &\Longleftrightarrow D \leq \frac{R}{\tan(n\cdot\theta/2)}. \end{aligned} \tag{3}$$

This inequality shows that an observer can see an object if the distance that separates them is less or equal to "$R/\tan(n\cdot\theta/2)$." This gives the maximal distance of perception under which the user sees the other avatar;

$$D_{\max} = \frac{R}{\tan(n\cdot\theta/2)} = R\cdot V, \quad \text{where } V = \frac{1}{\tan(n\cdot\theta/2)}. \tag{4}$$

This maximal distance depends on the player's viewing capacity V and on the observed avatar's size R.

The number of visible avatars is directly related to the player's viewing capacity. However, it is possible that at a certain moment during the game, the system or a user becomes no longer able to handle more message transfers. To prevent this, we allow it to artificially reduce the values of perception distance by using the filtering coefficient "FC" that varies between 0 and 1:

$$D_{\max} = R\cdot V\cdot \mathrm{FC}. \tag{5}$$

4.2. Filtering mechanism

To be able to use perception-based filtering, a server (in client/server architectures) or clients (in peer-to-peer architectures) compute the maximal distance of perception that the observing avatar has of the observed avatar (a simple multiplication). Then, they compare this maximal distance of perception to the distance that separates the two avatars to deduce if the perception takes place or not.

5. DETERMINING RADIUSES OF INTEREST ZONES

As we have seen in Section 2, many interest management techniques use zones to determine avatars' interests and manifestations. Until now, these techniques have determined the radiuses of their zones empirically. In this section, we propose mechanisms that determine these radiuses based on our perception rules. This translation has two goals: providing a mechanism to determine the size of zones for these used filtering techniques and establishing a fair ground to compare the perception-based filtering with the existing interest management techniques.

The condition that the zones should strictly fulfil to express perception is that whenever an avatar sees another with perception-based filtering, it should be able to see it with the parallel interest management techniques (area of interest management, spatial model of interaction or effect management, etc.). If this entails a reception of unnecessary information, this can be tolerated. But not receiving important update messages cannot be tolerated.

In Figure 4, we can see the translation of perception rules into various interest management techniques. The computation of these radiuses varies according to the techniques, but it always depends on the perception and perceptibility of all the avatars of the environment. In the following, we analyze these dependencies and their impact on filtering.

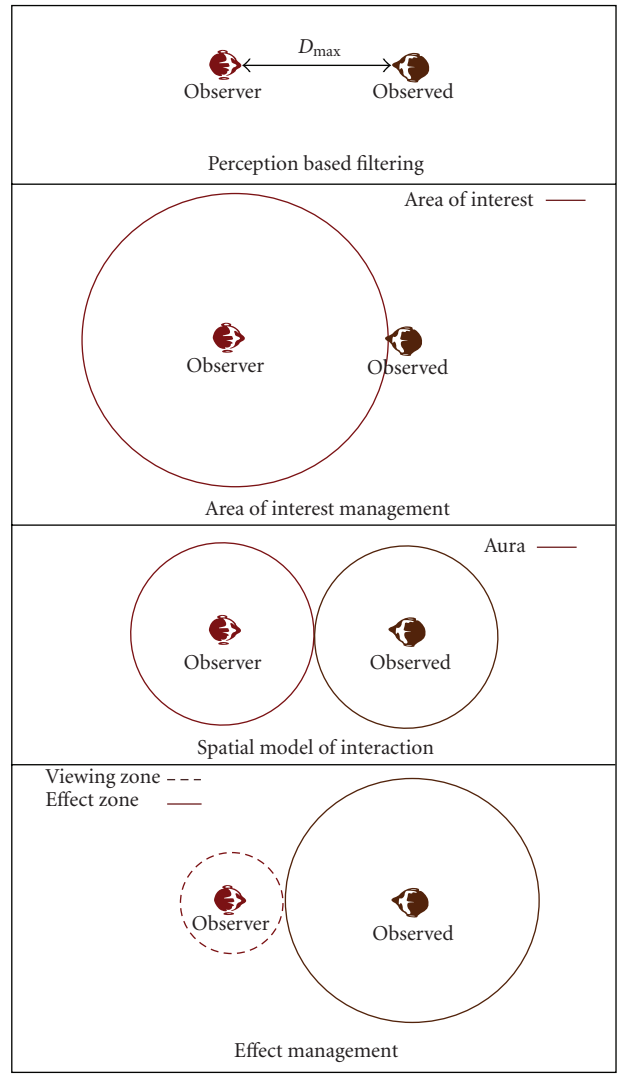


Figure 4: Translation of perception rules in various interest management techniques.

5.1. Area of interest management

The area of interest management allows an avatar to see the avatars that are within its area of interest (AOI). To have a fair translation, whenever an avatar sees another with perception-based filtering, it should see it with area of interest management. Therefore, if the distance between two avatars is smaller than $D_{\max}$, the AOI of the viewing avatar should cover the other avatar (see Figure 5). Consequently, the radius of the AOI should be greater than or equal to all the $D_{\max}$. To optimize filtering, we choose the largest $D_{\max}$ that the avatar has as its AOI radius.

S being the set of games' avatars:

$$\mathrm{AOI}(A) = \max\{D_{\max}(A, n),\ \forall n \in S\}. \tag{6}$$

This may entail excessive information reception because if we have an important avatar, all the avatars will enlarge their AOIs to be able to see it when it is distant. This will make

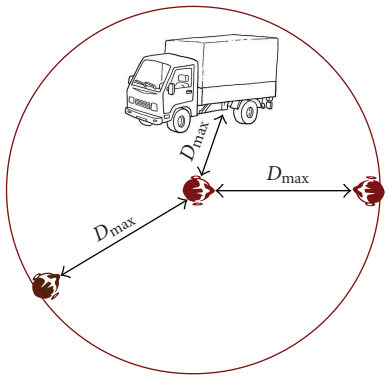


Figure 5: Determining the AOI radius.

the avatars receive information of small and distant avatars that they are not supposed to see.

5.2. Spatial model of interaction

The spatial model of interaction allows two avatars to see each other when their auras collide. The sizes of the auras will be defined following the rule **R**.

"A sees B with perception-based filtering" means that the distance between A and B is less than or equal to the corresponding $D_{\max}$:

$$D(A,B) \leq D_{\max}(A,B). \tag{7}$$

And "A sees B with the spatial model of interaction" means that the auras of A and B collide:

$$D(A,B) \leq \text{Aura}(A) + \text{Aura}(B), \tag{8}$$

where $\text{Aura}(N)$ is the radius of N's aura.

A condition that makes **R** possible is

$$\mathbf{C}\text{: } \text{Aura}(A) + \text{Aura}(B) \geq D_{\max}(A,B) \tag{9}$$

because

$$\begin{aligned} D(A,B) &\leq D_{\max}(A,B), \\ D_{\max}(A,B) &\leq \text{Aura}(A) + \text{Aura}(B) \Longrightarrow D(A,B) \\ &\leq \text{Aura}(A) + \text{Aura}(B). \end{aligned} \tag{10}$$

So condition **C** allows the satisfaction of rule **R**. Therefore, S being the set of the game's avatars, the general rule will be

$$\forall a \in S,\ \forall b \in S, \quad \text{Aura}(a) + \text{Aura}(b) \geq D_{\max}(a,b). \tag{11}$$

This rule gives a set of constraints which define a system of inequalities. For example, if we take a game with three avatar types: a person (P), an eagle-eyed person (EE), and a truck (T), the following constraints will be raised:

$$2{*}\text{Aura}(P) \geq D_{\max}(P,P), \tag{12}$$

(to enable two persons to see each other when they should),

$$\text{Aura}(P) + \text{Aura}(EE) \geq D_{\max}(P,EE) \tag{13}$$

(to enable a person to see an eagle-eyed person),

$$\begin{aligned} \text{Aura}(P) + \text{Aura}(T) &\geq D_{\max}(P,T), \\ \text{Aura}(EE) + \text{Aura}(P) &\geq D_{\max}(EE,P), \\ 2 \times \text{Aura}(EE) &\geq D_{\max}(EE,EE), \\ \text{Aura}(EE) + \text{Aura}(T) &\geq D_{\max}(EE,T), \\ \text{Aura}(T) + \text{Aura}(P) &\geq D_{\max}(T,P), \\ \text{Aura}(T) + \text{Aura}(EE) &\geq D_{\max}(T,EE), \\ 2 \times \text{Aura}(T) &\geq D_{\max}(T,T). \end{aligned} \tag{14}$$

These constraints define a system of linear inequalities which has an infinity of solutions. Our goal is to find the optimal solution that minimizes the number of possible interactions between avatars. This number is proportional to the possibility of auras' collisions, which makes it proportional to the total area covered by all avatars' auras:

$$AA = \sum_{n \in S} \pi\, \text{Aura}(n)^2. \tag{15}$$

Such a system can be solved in many ways. We used Newton's method for the solution of systems of equalities and inequalities [13], parameterized to minimize the value of AA.

5.3. Effect management

Effect management offers better precision because for each avatar we distinguish the ability to see (perception) from the ability of being seen (perceptibility), this gives more precise constraints. A similar demonstration to that of the spatial model of interaction gives the following translation condition:

$$D(A,B) \leq V(A) + E(B), \tag{16}$$

where $V(A)$ is the radius of an avatar A's viewing zone and $E(B)$ is the radius of an avatar's B effect zone.

So, S being the set of the game's avatars, the system takes this form:

$$\forall a \in S,\ \forall b \in S, \quad E(a) + V(b) \geq D_{\max}(a,b). \tag{17}$$

If we take our previous example, we will have the following system of inequalities:

$$V(P) + E(P) \geq D_{\max}(P,P), \tag{18}$$

(to enable persons to see each other)

$$V(P) + E(EE) \geq D_{\max}(P,EE), \tag{19}$$

(to enable a person to see an eagle-eyed person)

$$\begin{aligned} V(P) + E(T) &\geq D_{\max}(P, T), \\ V(EE) + E(P) &\geq D_{\max}(EE, P), \\ V(EE) + E(EE) &\geq D_{\max}(EE, EE), \\ V(EE) + E(T) &\geq D_{\max}(EE, T), \\ V(T) + E(P) &\geq_{\max} (T, P), \\ V(T) + E(EE) &\geq D_{\max}(T, EE), \\ V(T) + E(T) &\geq D_{\max}(T, T). \end{aligned} \tag{20}$$

The system can be solved by minimizing the area covered by the effect and viewing zones:

$$\text{AEV} = \sum_{n \in S} \pi C(n)^2 + \pi E(n)^2. \tag{21}$$

6. EXPERIMENTAL RESULTS

To implement perception-based filtering, we used Architecture for Systems of Simulation and Training in Teleoperation (ASSET) [14], a prototyping system supporting the design and evaluation of new teleoperation systems by using virtual reality components. In our context, ASSET is used in simulation mode only thus becoming a generic client/server-based DVE system.

We compared the performance of perception-based filtering with those of area of interest management and spatial model of interaction in its simple version and effect management. The level of realism is constant with the four techniques, since it is fixed by the perception rules while the communication load varies following the filtering technique and this is what interests us.

6.1. Experimental environment

We used for our experiments 100 nodes of a cluster of Grid'5000 [15], which is a grid of thousands of CPUs distributed at 9 sites France wide. Nodes have Rocks Linux 3.3.0 as operating systems and are connected by a Gigabit Ethernet network. The nodes are *IBM eServer 325* shipped with AMD Opteron 246 CPUs (2.0 GHz/1 MB/400 MHz), 2 GB memory, 2x Gigabit Ethernet (1 in use) and Myrinet-2000 (M3F-PCIXF-2) network cards.

An experiment is a two-minute simulation with a simulation step of 200 milliseconds. We carried out the experiments varying the number of clients. We repeated the experiments several times and averaged the resulting values. During an experiment, the server collects various experimental data such as the number of exchanged messages and the server's load.

The server was always allocated a dedicated node. However, since the number of clients in a single simulation got up to 160 (and we only have 100 nodes available), we eventually had to run two clients on the same node.

6.2. Simulation parameters

The virtual environment is a 20000 × 20000 square. The behavior of avatars is autonomous following a random acceleration with a privileged direction. An avatar can be a normal person, an eagle-eyed person (with good visual ability) or a truck. The person and the driver of the truck have a limited viewing capacity, while the eagle-eyed person has an enhanced viewing capacity. Furthermore, the truck has a larger size, thus it has a stronger chance of being seen than the others.

6.2.1. Perception-based filtering

For perception-based filtering, we consider that the field of view is equal to 120 degrees and the screen is 1024 pixels large. This gives us the angle covered by one pixel,

$$\begin{aligned} \theta &= \frac{\text{fov}}{Nb\,\text{Pixels}\,X} \\ &= \frac{120}{1024} \\ &= 0.117. \end{aligned} \tag{22}$$

We consider that the eagle-eyed person is able to see the details on the screen up to one pixel ($n = 1$). His viewing capacity is of

$$\begin{aligned} V_{\text{eagle}} &= \frac{1}{\tan}\left(n_{\text{eagle}} \cdot \frac{\theta}{2}\right) \\ &= \frac{1}{\tan}\left(1 \cdot \frac{0.117}{2}\right) \\ &= \frac{1}{\tan}(0.058) \\ &= 977.8. \end{aligned} \tag{23}$$

To obtain the maximal distance of perception that the avatar has of a person, we multiply the radius of the person with the visual capacity of the avatar:

$$\begin{aligned} D_{\max}(\text{eagle,person}) &= R_{\text{person}} \cdot V_{\text{eagle}} \\ &= 1.022 \cdot 977.8 \\ &= 1000. \end{aligned} \tag{24}$$

On the other hand, the maximal distance of perception that an eagle-eyed person has of a truck depends on the truck's bounding sphere radius:

$$\begin{aligned} D_{\max}(\text{eagle,truck}) &= R_{\text{truck}} \cdot V_{\text{eagle}} \\ &= 2.045 \cdot 977.8 \\ &= 2000. \end{aligned} \tag{25}$$

The avatars with a normal viewing capacity (the person and the driver of the truck) will see less clearly on the screen.

Table 1

D_{max}	Person	Truck	Eagle-eyed
Person (P)	500	1000	500
Truck (T)	500	1000	500
Eagle-eyed (EE)	1000	2000	1000

Table 2

Avatar	Person	Truck	Eagle-eyed
AOI	1000	1000	2000

Table 3

Avatar	Person	Truck	Eagle-eyed
Aura	250	1000	1000

Table 4

Avatar	Person	Truck	Eagle-eyed
Viewing zone	250	250	1000
Effect zone	250	1000	250

We will consider that their minimal threshold of perception is of two pixels ($n = 2$). Thus, we have

$$\begin{aligned} V_{\text{person}} &= \frac{1}{\tan}\left(n_{\text{person}} \cdot \frac{\theta}{2}\right) \\ &= \frac{1}{\tan}\left(2 \cdot \frac{0.117}{2}\right) \\ &= \frac{1}{\tan}(0.117) \\ &= 488.9. \end{aligned} \tag{26}$$

This influences all the maximal distances of perception of the person.

Table 1 lists the maximal distances of perception of our environment. We can see for example that the maximal distance of perception that a person has of a truck is 1000.

6.2.2. Area of interest management

By using the rules of translation defined above, we defined the radiuses of the areas of interest for our avatars in Table 2.

6.2.3. Spatial model of interaction

The system of inequalities related to the spatial model of interaction was solved using a solver which gave the results listed in Table 3.

6.2.4. Effect management

Table 4 shows the optimal solution of the corresponding system of inequalities.

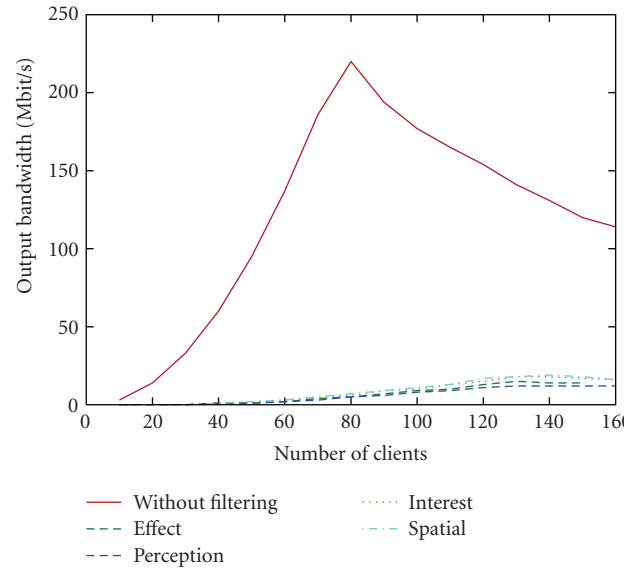


Figure 6: Network bandwidth with and without filtering.

6.3. Results

6.3.1. The impact of filtering

Figure 6 shows the rate of messages sent by the server with and without filtering. We notice that all the filtering techniques succeeded in reducing the rate of outgoing messages. This pushes further the bottleneck on the server that is due to the cost of sending messages when there is no filtering used. However, filtering techniques postpone the bottleneck but do not eliminate it. The bottleneck is related to the load of filtering that is equivalent in all the filtering techniques (Figure 7).

We should note that we use a Gigabit network. If the network had a narrower bandwidth, the bottleneck without filtering would take place a lot sooner because of the network load and not the server load.

6.3.2. Evaluation of perception-based filtering

Figure 8 compares the rate of messages sent by the server with the four filtering techniques. We can notice that the perception-based technique presents the lowest rate of sent messages. This exact filtering performed by our technique allows the server to use less upload bandwidth and makes users receive fewer messages. This makes the application less expensive and more interactive.

6.3.3. Analysis

We have seen that perception-based filtering achieves better results than the other filtering techniques. This is due to the fact that perception-based filtering performs an exact filtering and delivers to each user only data that he is able to perceive while others do not.

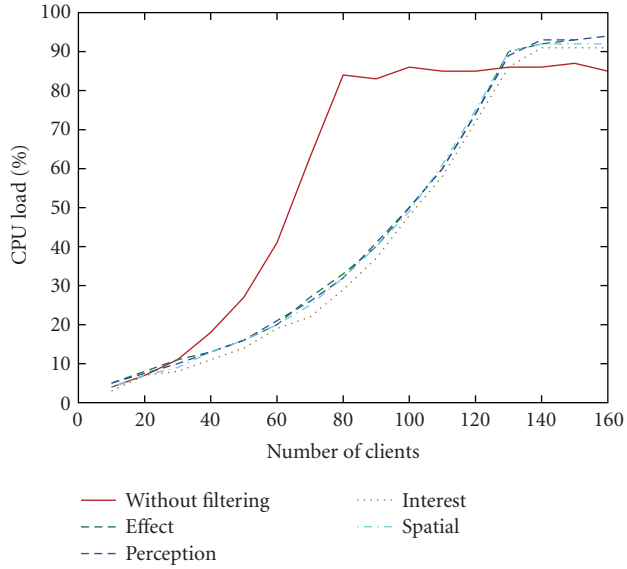


Figure 7: Server's load with and without filtering.

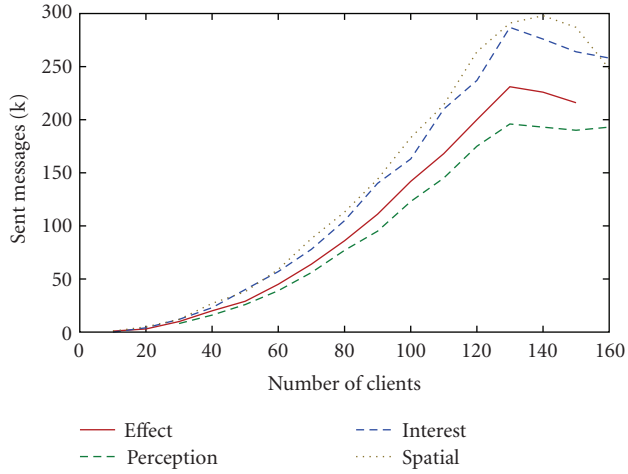


Figure 8: Sent messages.

(i) The area of interest management does not allow avatars to express their manifestations. For these reasons, the AOI of an avatar depends on all the existing avatars. Therefore, if we have an important avatar that is seen from a long distance, the AOIs of all the avatars should be large enough to cover this avatar when it is distant. Thus, these AOIs will cover small and distant avatars that should not be seen.

(ii) The spatial model of interaction combines the ability to see and the ability to be seen within the aura. This prevents avatars from establishing asymetrical relations. Therefore, an important avatar which is seen by everyone will be forced to see everyone.

(iii) Effect management does not suffer from the shortcomings of the previous techniques. That is why it offers a better rate of filtering than both of them. However, it still has an excessive rate of message exchange due to the sizes of effect and viewing zones which depend on each other because of the system of inequalities.

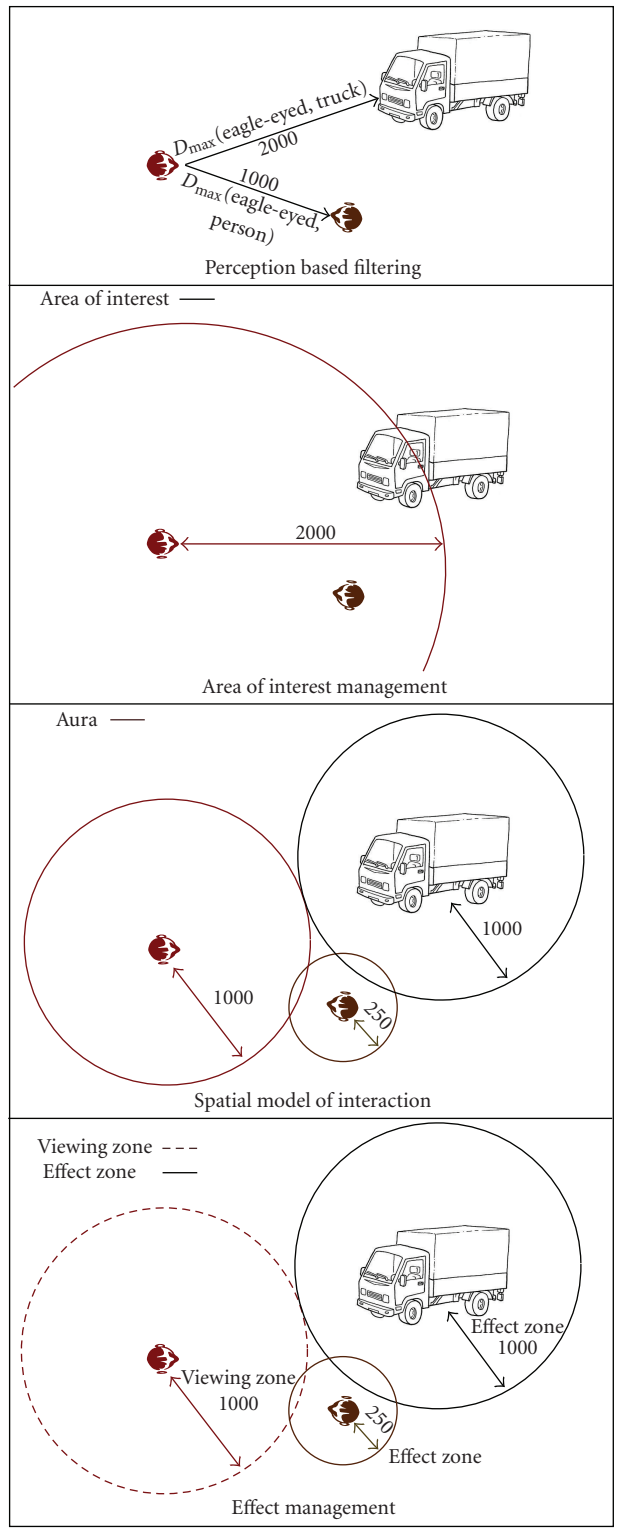


Figure 9: D_{max} translation into various interest management techniques.

Figure 9 gives a concrete example of the inexact filtering.

Let us consider the scenario of an eagle-eyed person who is standing at a distance of 2000 away from a truck and of

1100 from a normal person. We will study the perception of the eagle-eyed toward the two other avatars.

With perception rules, our observer can see the truck because they are at a distance (2000) which is equal to the observer's maximal distance of perception of the truck. However, the distance that separates him from the other person (1100) is superior to the corresponding maximal distance of perception (1000). Therefore, the eagle-eyed person receives the update of the truck but not that of the other person.

With area of interest management, the observer's AOI covers the truck and the person. So the observer will receive the update of the person, while he is not supposed to see him.

With the spatial model of interaction, the aura of the observer overlaps the auras of the two other avatars. Thus, he receives the messages of the other two avatars.

With effect management, the viewing zone of the observer overlaps the effect zones of both the truck and the person. So the observer also receives unnecessary updates.

This concrete comparison further details the causes of the advantage of percepetion-based filtering over the other interest management techniques.

7. CONCLUSIONS AND FUTURE WORK

We have presented the perception-based filtering technique. This technique will help developers determine their avatars' perceptual interests in a precise way. We have also presented a translation of perception rules into interest management techniques which define the interests of players by using zones. We have then compared the performance of perception-based filtering with other techniques and we found that our technique further reduces the amount of exchanged messages. We have also described the reasons why our technique offers such advantages.

Perception-based filtering presents an additional advantage related to the dynamic change of interests. In fact, perception-based filtering allows avatars to change their abilities to perceive or be perceived without having to change a lot of simulation parameters. Furthermore, any change of ability of one avatar with the other techniques will induce a new system of inequalities that should be solved to give a new set of filtering parameters.

However, perception-based filtering only expresses perceptual factors while the other techniques allow to express other characteristics in addition to perception (function, behaviour, etc.). This is why we intend to extend our research to the auditive and functional domains.

ACKNOWLEDGMENTS

This work was supported in part by the French Government through the NatSIM ANR Project (Contract ANR 05-MMSA-0004-01). Grateful acknowledgment for proofreading and correcting our article goes to Anne Beauvallet, Senior Lecturer at Toulouse II University (English Studies Department).

REFERENCES

[1] D. Terdiman, "World of warcraft battles server problems," http://articles.techrepublic.com.com/2100-1087911-6063990.html.

[2] S. Elmerhebi, J.-C. Hoelt, P. Torguet, and J.-P. Jessel, "Perception based messages filtering for massively multiplayer online games," in *Proceedings of the 3rd International Conference on Games Research and Development (CyberGames '07)*, Manchester, UK, September 2007.

[3] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham, "Exploiting reality with multicast groups: a network architecture for large-scale virtual environments," in *Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS '95)*, pp. 2–10, Los Alamitos, Calif, USA, March 1995.

[4] S. Benford and L. Fahlén, "A spatial model of interaction in large virtual environments," in *Proceedings of the 3rd European Conference on Computer-Supported Cooperative Work (ECSCW '93)*, pp. 109–124, Milano, Italy, September 1993.

[5] S. Elmerhebi, P. Torguet, and J.-P. Jessel, "Realism and communication evaluation of effect management in distributed virtual environments," in *Proceedings of the 12th Eurographics Symposium on Virtual Environments (EGVE '06)*, Lisbon, Portugal, May 2006.

[6] A. Boukerche, N. J. McGraw, C. Dzermajko, and K. Lu, "Grid-filtered region-based data distribution management in large-scale distributed simulation systems," in *Proceedings of the 38th Annual Symposium on Simulation (ANSS '05)*, pp. 259–266, San Diego, Calif, USA, April 2005.

[7] J. Calvin and R. Weatherly, "An introduction to the high level architecture (HLA) runtime infrastructure (RTI)," in *Proceedings of the 14th Workshop on the Standards for the Interoperability of Defence Simulations*, pp. 705–715, Orlando, Fla, USA, March 1996.

[8] R. C. Waters, D. B. Anderson, J. W. Barrus, et al., "Diamond park and spline: social virtual reality with 3D animation, spoken interaction, and runtime extendability," *Presence*, vol. 6, no. 4, pp. 461–481, 1997.

[9] T. A. Funkhouser, "Ring: a client-server system for multi user virtual environments," in *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '95)*, pp. 85–92, ACM Press, Monterey, Calif, USA, April 1995.

[10] D. Bartz, M. Meißner, and T. Hüttner, "Extending graphics hardware for occlusion queries in opengl," in *Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 97–104, ACM Press, Lisbon, Portuga, August-September 1998.

[11] J. M. Airey, J. H. Rohlf, and F. P. Brooks Jr., "Towards image realism with interactive update rates in complex virtual building environments," in *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '90)*, pp. 41–50, ACM Press, Snowbird, Utah, USA, March 1990.

[12] M. Abrash, "Inside quake: visible surface determination," in *Dr. Dobb's Sourcebook*, pp. 41–45, January-February 1996.

[13] B. N. Pshenichnyi, "Newton's method for the solution of systems of equalities and inequalities," *Mathematical Notes*, vol. 8, no. 5, pp. 827–830, 1970.

[14] N. Rodriguez, J.-P. Jessel, and P. Torguet, "A virtual reality tool for teleoperation research," *Virtual Reality*, vol. 6, no. 2, pp. 57–62, 2002.

[15] F. Cappello, F. Desprez, M. Dayde, et al., "A large scale, reconfigurable, controlable and monitorable grid platform," in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing (Grid '05)*, Seattle, Wash USA, November 2005.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 619108, 8 pages
doi:10.1155/2008/619108

Research Article

A Study of Interaction Patterns and Awareness Design Elements in a Massively Multiplayer Online Game

Tiffany Y. Tang, Cheung Yiu Man, Chu Pok Hang, Lam Shiu Cheuk, Chan Wai Kwong, Yiu Chung Chi, Ho Ka Fai, and Sit Kam

Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong

Correspondence should be addressed to Tiffany Y. Tang, cstiffany@comp.polyu.edu.hk

Received 28 September 2007; Accepted 13 December 2007

Recommended by Kok Wai Wong

Massively multiplayer online games (MMOGs) have been known to create rich and versatile social worlds for thousands of millions of players to participate. As such, various game elements and advance technologies such as artificial intelligence have been applied to encourage and facilitate social interactions in these online communities, the key to the success of MMOGs. However, there is a lack of studies addressing the usability of these elements in games. In this paper, we look into interaction patterns and awareness design elements that support the awareness in *LastWorld* and *FairyLand*. Experimental results obtained through both in-game experiences and player interviews reveal that not all awareness tools (e.g., an in-game map) have been fully exploited by players. In addition, those players who are *aware* of these tools are not satisfied with them. Our findings suggest that awareness-oriented tools/channels should be easy to interpret and rich in conveying "knowledge" so as to reduce players-cognitive overload. These findings of this research recommend considerations of early stage MMOG design.

1. INTRODUCTION

Interacting online with people from throughout the world is a daily occurrence for millions of Internet players, yet most do it with little perspective on the virtual identity they are projecting. It is generally accepted now that the Internet and online games provide a tremendous opportunity for new forms of entertainment. It is especially true for MMOG characterized by its ability to enable thousands of players to play in an evolving virtual world at the same time over the Internet. Once a player enters the game world, he can engage in a variety of activities with other players who might be sitting at the other part of the globe. Hence, one of the most foremost goals of these MMOGs is to offer a rich social platform for players to interact and socialize as Will Wright, creator of the block-buster game "The SIMS," put it: "*In some sense, what we are really building with these games are communities. That is our primary thing*" [1]. In this regard, awareness is known as one of the most discriminating factors contributing to the success of the social environments. It is defined as "*the knowledge of the presence of other people, including their interactions and other activities*" [2]. Generally, being aware of each other's presence (including the workspace environment, their actions, and the manipulating artifacts) provides a clue for their own action in the situated environment, and might guide their own actions accordingly [3]. It is especially imperative in densely populated online virtual communities, where people tend to interact with each other to weave a rather complex, yet, fruitful web of relationship [4]. Careful incorporation of awareness tools in these online spaces thus becomes more essential to foster both collaboration and competition. In the human-computer interaction area, a number of works have been devoted to study awareness, including how to make various awareness tools in a wide variety of applications [3, 5–7] and in MMOGs [5, 8–10]. In fact, MMOGs have attracted more attention recently as a test bed to study awareness and social interaction patterns in some high profile CHI conferences [5, 9]. Unfortunately, very few of them probe into this issue from players' perspective, that is, whether or not, players have made the most out of these tools to facilitate the in-group and interpersonal interactions, which motivates our study here. Particularly, in this paper, we report out findings on one popular MMOG called

Table 1: A brief summery of awareness and interaction design elements in *LastWorld*.

		Tools/approaches/tasks	Information elements
Social awareness	People	Player description (Figure 2)	Who is online?
			What are their statuses?
			What are they doing?
	Location	Map, radar (Figure 3)	Where am I?
			Where are others?
Social interaction		Teamworks such as chatting, fighting, trading	How can I team up with other players?
			How can I boost up my skills and levels in teams?

the *LastWorld* (available at http://www.lastworld.com/) and compare our results with that of *Fairyland*, on evaluating the usability of these tools to raise awareness and encourage players to interact.

In the next section, we will discuss previous study on MMOGs, with a particular focus on how MMOGs act as social sphere. We will then give brief overview *LastWorld* including how they have been designed to raise social awareness and encourage group interactions. We then report the experimental results on *LastWorld* in Section 3. A comparison of *LastWorld* and *Fairyland* on the design efforts to support social interactions will be provided in Section 4; the design implication will also be pointed out. We conclude this paper by discussing our next step in augmenting our research following the path.

2. MMOGS AS A "SOCIAL SPHERE"

2.1. Social awareness: some background

In a distributed, socially populated environment, it is imperative for members to be "*aware*" of each other and the environment, in terms of their many facets, among them, the information about each other's actions, the individual environment, and the state of the manipulated artifacts. For instance, in distributed document editing environment, this awareness information can range from who are the active editors, on what part of the document each are working, why the document is being edited, and so on. Game designers have implemented a variety of awareness tools to allow players to formulate general as well as specific awareness of their group mates, or even counterparts in an attempt to execute their actions accordingly. For example, a map can show player position, while the name on top of players reveals player identity to others. The work in [10] summarizes the workspace awareness elements and categorizes them into two major types based on their temporal aspect: those related to the *present* and the *past*.

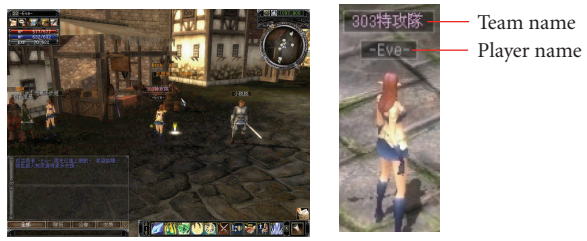
2.2. Motivation of our work

One notable issue related to awareness is that the tools to support awareness should be *readily accessible*, *easy to interpret*, and *rich* in convey "knowledge" in order to reduce players' cognitive information overload. It is especially true in real-time multiuser virtual environment like MMOGs, where players rely heavily on the information to explore, when there might be overcrowded information available on the screen. For instance, imagine a group of players collaboratively engage in a fighting mission at a remote island; valuable information related to it includes individual players' skills, energies left, location, and identifying the approaching enemies and so on, as shown in Table 1. Hence, when the information becomes overloaded, players have to quickly identify those valuable or in some cases, they even need to choose from among the various awareness sources, in order to gather this information and make a quick assessment of the environment and situations. One extreme is that the information might be too coarse, thus, cannot be used instantly to assist players. The other extreme is that there might be too much information which makes players difficult to spot the most relevant to their current task. Either of these two cases can greatly affect players' perceptions of the environments. To our knowledge, very few studies addressed this issue, which motivates our study here. The findings of our in-game experiences and player interviews agree with our worries in that players, in fact, are not satisfied with the awareness tools, and pointed out that they are sometimes too busy to compile the information and formulate their presence in relation to the environment and other players.

Before we proceed to present our findings, a discussion on related work will appear in the next section.

2.3. MMOGs as a "Social Sphere"

MMOGs are designed to encourage players socialize through a wide variety of channels, from combating, gesturing, chatting, doing business, and so on; a collected place where we call it a social sphere. Although MMOGs have taken the game-playing world by storm, the work in [8] pointed out that there are lacks of sociological study in the research community. One of the most notable studies was conducted by Ducheneaut and Moore [5], where the researches immersed themselves in Star Wars Galaxies (SWGs), one of the most popular MMOGs to investigate the interaction patterns to support CSCW. The work in [8] further investigates the degree of social activities as supported and exhibited in the "*third places*" of SWGs: the cantinas. In particular, as originally coined by Oldenburg, these "*third places*" should



(a) (b)

FIGURE 1: (a) A "day" in the world, (b) one of the people awareness tools.

(a) (b)

FIGURE 2: (a) People awareness: who's online?; (b) people awareness: where is the player and who is he/she?.

provide "*a great variety of public places that host the regular, voluntary, informal, and happily anticipated gatherings of individuals beyond the realms of home and work*" [11, page 16]. The works in [5, 8] reveal that the majority of visitors have clear *purposes* to socialize in the cantinas, which is in contrast to the pure social interactions that go "*beyond the contexts of purpose, duty or role*" [12] and their encounters are mostly marked by "*short and instrumental*" [8, page 11]. To summarize, these studies [5, 8–10] generally examine a variety of social activities players carried out during game-playing to, in some degree, look into the design rationale that MMOGs should encourage and support social interactions. The work in [13] reviews a number of awareness tools in a video game, *Quake*, to investigate how these tools can support team play and team collaboration. However, the study did not reveal players' perceptions on these tools, which is one of the major differences between their study and ours.

Although in these studies, awareness issue has been casually mentioned, it is not thoroughfully investigated, which motivates our study here. Specifically, we attempt to study how sufficient existing awareness tools have been designed to foster social interactions among players, and whether or not players tend to make the most out of these tools to engage in social interactions.

Before we proceed to present our findings, we will give a short overview of *LastWorld*.

2.4. Some background on LastWorld

LastWorld was launched in the summer of 2005 and rated as the number online game at the time of this study [14]. As an MMOG, it aims at providing an entertaining and social platform for players. In order to support social interactions, *LastWorld* provides a wide range of tools, tasks to allow players to interact with each other, as well as with non-player characters (NPCs). Figure 1(a) shows one screenshot of the game, where players can control their characters to take actions, such as fighting, sitting, working, trading items, and so on. Players can also communicate with each other, choosing different types of communicational methods through the conversational panels in the lower right corner of the screen, as shown in Figure 1.

2.4.1. Social awareness in LastWorld

A number of awareness tools exist in the games. In this paper, we will only focus on tools to support *people* and *location awareness*.

Table 1 summarizes the tools/tasks that we will evaluate in *LastWorld* that are designed to support social awareness and interactions.

Regarding people awareness, the game uses a series of tools to indicate the *status* and *identity* each player. The purpose of these tools is to help users to be aware of other players' position as well as obtain their identities/skills.

For example, above each character, there are names showing in white and the organization name showing in pale purple, as seen in Figure 2(b). And on notice board, once a team member is connected to the game, it will notify other members of the same team that "[who] member is connected." Players can also check whether or not their friends are online instantly in the conversation box (see Figure 2(a)).

Location awareness deals with the information players can collect related to whereabouts of themselves, other players, monsters, and NPCs. Sufficient and easy-to-obtain location awareness information can give players an orientation as well as the position of others that can help them formulate their activities and achieve their goals. *LastWorld* provides a number of tools, including *map*, *radar*, *coordinates* to indicate location of players, players' friends, players' teammates, enemies, and so on. Players can manipulate it by zooming in and out on it, and perform searching (see Figure 3).

2.4.2. Social interactions in LastWorld

Both player-player and player-game interactions are encouraged in *LastWorld*; the former includes a number of communication channels for players to interact with each other; while the latter provides ways for players to interact with NPCs. In this paper, we focus on the former type of the interactions (see Table 1).

The game provides a couple of communication channels for players to interact with each other. For instance, in

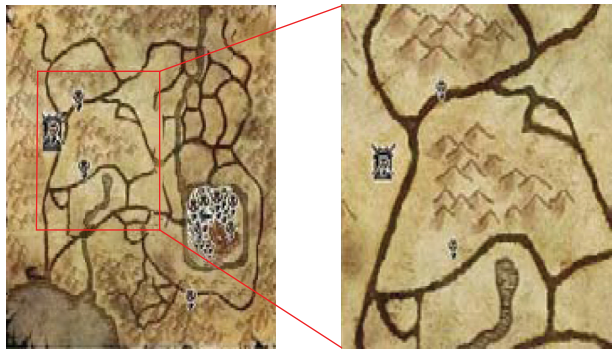


Figure 3: Normal map versus zoom-in map.

Trading, players can buy/sell goods in fixed price or by bargaining in any city. Both buyers and sellers can negotiate the prices during trading. In general, Trading provides an excellent platform to encourage players with different professions and skills to communicate with each other. *Chatting* offers another type of player-player interaction in the game where players can engage in 4 types of chatting mode: *public, private, team,* and *organization*.

3. FINDINGS ON *LASTWORLD*

3.1. Experiment methodology

To have a real experience in the game community, each of us created at least one character in the game. A total of 10 characters were created, which make up most of the professions in the game. In order to become a part of the community, each of the characters logs on to play the game for at least 40 hours. We then collect data through direct observation of the game environment and interacting with different players in the game. These interactions include participating in different teams, trading, and bargaining with other players. In total, we spent 2 months playing the game regularly and after that, we designed a questionnaire for the team members to complete. The questionnaire targets at how each group of players looks at the awareness tools, their interactions, their attitude, and behavior in the game. We then further interviewed 18 players (with their age ranging from 20 to 30) in the game so as to increase the reliability of the collected data. We tried to divide the players according to their playing frequency. The results are sufficient for us to address the issues raised in this paper.

3.2. How players socialize in LastWorld: experiments

As argued in previous sections, the most fundamental goal of MMOGs is to create a social environment for players to interact through all the possible kinds of awareness-oriented design elements and interaction channels. These elements include a variety of designs in the game, including map, radar, chat, and so on (see Table 1 for those studied in our experiment). We are interested in investigating the degree of awareness of players; that is, how efficiently and easily players can make use of these elements to foster social interaction, and improve their in-game social welfare. In particular, the following questions are addressed.

(i) do players know and use these carefully designed awareness elements during game playing?
(ii) how efficiently the awareness tools can foster players' social interaction?

The first goal is to evaluate the usage of these virtual awareness tools, while the 2nd is to assess the usability of these tools in facilitating player interactions. In our experiment, we studied three types of players, that are, core players, spending more than 30 hours per week; moderate players, spending 11 and 20 hours per week, and casual players, spending less than 11 hours per week. In our study, there is 6, 8, and 4 core, moderate, and casual players, respectively. It is crucial to include all types of players in the analysis so as to understand and compare different players' views on awareness and their interaction.

3.2.1. The usefulness and appropriateness of awareness design in LastWorld

We mainly used questionnaire to gather the information about the awareness for our study. We found out that players have similar points of view in some aspects, but have different opinion in others.

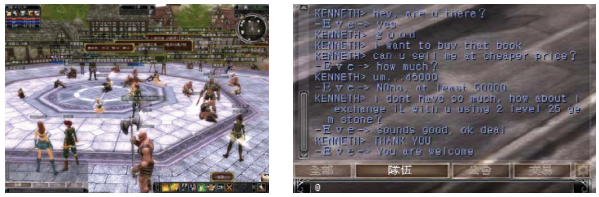
(a) People awareness—Are players aware of the social environment around them?

Issue One. How long does it take for players to be aware that their friends are online?

The result of our study reveals that the majority of players are able to know the status of their friends immediately or within 3 minutes. This suggests that the notification of the friends' connection is helpful for the players who are notified whenever their friends are online. However, 5 out of 18 players are unable to determine whether or not their friends are online. There are some possible reasons for it. Since the notice board shows many things concurrently, newer messages will cover the older ones, so players may easily miss some notification in the board. In addition, players may be busy doing something else such as hitting monsters in the game, so they fail to pay attention to the notice board. As such, some sound alert could be used; and it is often referred to as audible awareness indicators [3].

Issue Two. How difficult is it to find friends?

Obviously, the textual descriptions on top of each character are essential for the players (see Figure 2(b)): at least half of them strongly agreed. The results indicate that the character, organization, and other descriptions are critical to identify players. Surprisingly though, roughly half players, 44%, find it difficult or very difficult to identify their friends from among other people, even though the friends are just next to them, because although these elements can help players identify characters, it is so confused and complicated especially when there are too many players showing on the screen at the same time. To find out how difficult to identify



(a) (b)

Figure 4: (a) The town centre on a particular day, and (b) some players are willing to talk during trading.

friends, we performed an experiment: we open the game together and sit in the crowd. One of us spent almost 3 minutes to locate all of "us." When asked what helped him to do so, he pointed to the radar and coordinates rather than player description over the head. Therefore, the helpfulness of player description is less useful if there are too many characters in the same place (see Figure 4(a)).

More than half of core players think it is easy or very easy to locate their friends. Most moderate players vote it as normal or difficult; while all casual players rated as easy or very easy. Core players have much experience in playing game, so they are more experienced to find other ways such as using radar and communication to identify the friends. No other tools might help players find other players in the game except for the description of the characters. For instance, if one player wants to find a personal shop to buy a weapon, what he/she can do is just running around the town and finding a player who is a weapon seller. If he/she wants to find a specific character type to form a team, he/she can only send a message to everyone and wait for their reply, which is very time consuming. In other words, the tools in the game are not sufficient to foster awareness among players.

(b) Location awareness—The usefulness of corresponding design elements

Issue One. Are maps, radars, and coordinates easy to use in support of location awareness?

56% of players think the tools are useful to support location awareness. Our study indicates that a large proportion of players think that map, radar, and coordinates are easy or even very easy to use. Also, 22% of them have neither stronger nor weaker views on this issue. Only one player feels negative with the maps and radars. There are no differences between the three types of player regarding it. This result is not surprising since the game provides comprehensive functions to assist players to use such tools.

Issue Two. Is there sufficient information in the map and radar?

Although the map and radar are useful for players, they may not contain enough information to help players: 78% of players pointed out that the information embedded in the map and radar is not clearly enough to facilitate higher degree of location awareness: know where they and their friends are. The results of three types of player are similar. To further our understandings, we interviewed 10 players in the game on it. 70% of them reported that there are not enough notations in the map and radar. And 20% of them think that it is just ok. Among the 7 players who admitted that notation is not enough, some pointed out that the map is not detailed enough. Some of players indicate that the details in the map cannot be changed when we zoom in the map except for the town of the current game, as shown in Figure 3 that even though the map is zoomed in, the detail of it will not be enhanced. In addition, the rest of players agree that the map's notation is not enough since the map has no coordinates when the cursor points to the map to help players find others. Some especially novice players point out that the coordinates are meaningless.

3.3. Interaction patterns in LastWorld

In another series of experiments, we attempt to study information on how players make use of the interactions provided by the game to enrich their socially virtual "life." In particular, we will focus on teamwork and trading system, two of the most representative interactions in the game.

3.3.1. TeamWork

Issue One. Have players ever tried to team up with other players? Are they willing to team up with other players, both in team or outside the team?

Generally, our experiment summarizes that players feel satisfied from working and collaborating with others. Among the 18 players we interviewed, about 90% of them have tried to be involved in different teams. Nearly all (except for one moderate player) expressed their willingness to team up with players of other professions. There are no apparent differences between three groups of players. Furthermore, about 90% players have tried to team up with other players, and only 2 of them are active in socializing with players out of their team. That is, most of the players are reluctant to talk actively with other players; they never or seldom talk actively with players outside their team. However, the number jumped to 15 (94%) when they got involved in the teams. The majority of them commented that they tend to talk more with their team members than with people outside the team.

Issue Two. Why and how players interact with each other through teamwork and in what ways?

The above result gives insufficient information on why and how the players interact through teamwork. To further our understandings, we turn to compare the different effects between players that always team up with other player and those that play the game alone. We created 2 similar characters, A and B, of the same profession and at the same time. A tends to team up with other players, while B is not involved in teamwork at all. We let the two characters do the same thing to gain experience within 5 hours. In the end, A achieved 15 levels, while B achieved 12 levels only. A died once only while B died 5 times. In addition, their interactions with other players are also very different. A always exchanges sentences and communicates with team members, while B

Table 2: A comparison of two players in a team and playing alone, respectively.

	Character A (Involved in a team)	Character B (play alone)
Game hour	5	5
Level achieved	15	13
Deceased times	2	5
Interaction (sentences exchange)	Around 180	Around 15

exchanged only a few sentences (roughly 15 sentences within 5 hours) with other players, as shown in Table 2.

These observations are aligned with our previous analysis: it explains why players tend to work and collaborate with other players. That is, those involved in the team tend to gain more experience and die less, which provides a strong intrinsic incentive for players to be involved in teamwork. In particular, the system awards 20 percent more experience to each player involved in the team and players can gain experiences much faster. Meanwhile, when there are characters from different professions involved in a team, the overall power of the team also becomes stronger. The more players involved in a team, the stronger the team becomes. It is not difficult to see that the team members have to have good teamwork so as to survive in more dangerous areas. They have to communicate with each other, and also due to the interdependencies of the jobs, they have to communicate in order to get helps from other team members. Another kind of relationship that evolves from the conversation is altruism among team members, through which players with high levels give a helping hand to new players. These various ways provide an ideal platform to encourage players to help and interact with each other. The more the ways in which players can and are encouraged to help each other, the easier it is for players to meet each other, which basically reinforces the relationships among team members and facilitates group interactions.

3.3.2. Trading

Issue One. Have players ever tried to trade?

Out of 18 players, 14 have tried before. All core game players have tried. This number dropped sharply to 2 (out of 4) for casual players. For those who have ever tried to trade, only a small fraction has tried to bargain with the buyers or sellers. The number is surprisingly low. The result shows short and casual interactions exchanged among players. We found out that for those 14 players that have ever tried to trade, 11 usually do not make any utterances during the trade at all. Though some of them exchanged a few sentences, none exchanged more than 5 sentences during the trade. The data shows that most players have tried to trade without many interactions.

Issue Two. How often do players talk during trading?

To answer this question, we set up 3 personal item shops in the town centre selling the same items. In store A, we set the price of all items 20 percent higher than the market price, while in store B, we set the price 20 percent lower; and to have a control experiment, we further set up store C with all items selling at the market price. After that, we let the stores be idle for 3 hours. No items in store A were sold, while in store B, 18 items were sold and in store C, 4 items were sold. However, for all cases, no players attempted to bargain or talk with us. Nearly all go to the personal item shops simply searching for what they need, without greeting or talking with the merchandiser. They buy items at a reasonable price, and leave nearly immediately when it is rather expensive. The town centre looks like very crowded, with high population density. The players sitting on the square with a text box above them are those opening personal item stores. We spent 30 minutes greeting 100 players. However, only a few of them gives us some response. A major cause is that many players are away from the keyboard (AFK), leaving their avatars idle to earn money for them. When this happens continuously, the players visiting the personal item shops and trying to bargain with the merchandisers will get frustrated. They sense that all players trading in the town centre are AFK-ing (or "microing") and avoid talking or talk very little. This also explains our previous data, most players tend not to bargain with the merchandisers or say very little. This, of course, greatly affects the quality of interaction [5]. While these game features such as microing allow a wider range of activities to be performed automatically, they can compromise the quality of social interactivity. Nevertheless, this does not prevent some players from talking during the trading. We noticed that there are some players who are genuinely interactive and eager to have longer conversations with others (see Figure 4(b)).

4. A COMPARISON BETWEEN *LASTWORLD* AND *FAIRYLAND*

So far, we observed that player performance is largely determined by how players can utilize the tools and other game elements such as in-game tasks, and exploit accordingly. Our observations indicate that not every tool has been fully understood or noticed by players, and not every group-oriented task is capable of *forcing* players to execute group activities: some choose to complete the task alone, while some are willing to form a group. A natural question to ask at this point is that what would happen if we conduct the analysis in another type of game? To answer it, we perform a similar usability study in a different type of game, *Fairyland* which is a fantasy MMOG more suitable for girls. It is different from *LastWorld*, a more action and strategy-packed MMOGs preferred more by boys. In *Fairyland*, players enjoy similar experiences like those in *LastWorld*, for instance, chatting with other players, forming teams to perform tasks such as fighting, trading, and so on. The major difference between the two games is that *Fairyland* is targeted at girl players, while *LastWorld* is targeted more at boy players, which leads to some unique design elements to encourage player interactions. For instance, in *Fairyland*,

(a) (b)

Figure 5: (a) The map, (b) player description.

there is a Family system, where players can join a family to interact, socialize, and participate in all kinds of activities. The core of the Family system is fundamentally different from the Fighting system in *LastWorld* and many other games in that the Family system is inherently more group-oriented, and is a metaphoric design mimicking human family system. Therefore, we expect that the interactivity in this module should be denser than that in other module in the game. To make a comparison, we will briefly report key findings and compare them to those in *LastWorld.*

4.1. Major findings and the comparisons

Besides in-game experiences, we again distributed questionnaires to players, and received 55 responses. 95% agree that map and coordinates are useful to unfold two major kinds of location information (see Figure 5(a)): buildings or paths around the players, and the general information in the location. Player coordinates are also shown on the small map, so most of them feel happy with the design to support location awareness. 84% of players agree that situational icon containing player description similar to those in *LastWorld* on players' head helps them know what others are doing in the game (see Figure 5(b)). In addition, players feel positive toward the audio alert attached to a location; that is, when a player enters different places, the background music will change immediately in order to alert players that they are staying in a different location now.

Among the various types of group-oriented tasks, the Family mode stands out: 91% reported experiences with it at least once, and among them, 80% like it very much. It is also observed that once players join a family, they always interact with their family members. Half of the subjects indicate that by joining a family, they cannot only interact and engage in more group-oriented activities such as fighting enemies, doing business, but also win real friendship outside the game. Players are quite comfortable with the interaction mode reflecting bindings among players both inside and outside the game. The result reveals the success of design element in *Fairyland* to encourage players to interact and socialize. This finding is different from that in *LastWorld* where players tend to talk with others in a team. As for chatting, our finding is similar to that in *LastWorld* that not many players tend to talk unless required; instead, they prefer to just send some emotional icons during communications which are deemed enough in most occasions (96%).

These key findings lead us to strongly believe that even though the genres and target players of the two games are different, player expectations and perceptions over the usability in awareness and interaction of the two games are similar: the tools should be designed to allow easy and quick access; the information should be easy to interpret and manipulate. As for the task design, the success of the Family mode in *Fairyland* highlights the importance of the task *per se* to encourage players to collaborate, as opposed to the teamwork mode in *LastWorld* where players seldom talk outside a team.

4.2. Discussions and design implications

Although macroing can automate performance actions and offer a more flexible options for players to continuously engaging in games without physically sitting in front of the games, it, somehow, compromises the quality of social interactivity among players [5]. We also suggest that the deployment of audio alert (as in *Fairyland*) which can quickly inform players of some key information such as who is online, or what is the player's newest status, instead of only changing the color of player names which are relatively difficult to notice as in *LastWorld*. One very interesting observation is that some players are reluctant to socialize when they do not need to, as in trading in *LastWorld*. However, the success of the Family mode in *Fairyland* highlights the importance of the task *per se* to encourage players to collaborate. The result indicates the growing trend and *degree of importance* of these "third place" (i.e., the shop in *LastWorld*; the Family in *Fairyland*) to host the "*voluntary and informal*" gatherings [11, page 16].

Although awareness tools and interaction patterns are quite prevalent in many MMOGs, little is known about how players perceive and utilize these tools, and exploit them. Our study aims at filling this gap. We suggest that it is rather essential for designers to test whether or not players can make the most out of these tools to realize designers' design rationale.

5. CONCLUDING REMARKS

In this paper, we look into the interaction patterns and the awareness design elements in *LastWorld*, and briefly compare the results with a different type of MMOG called *Fairyland*. The results generally reveal that the emphasis on teamwork is one of the most important components to keep players interacting in the game, such as the success of the Family in *Fairyland*. Thus the MMOGs should include a variety of team-oriented activities and offer a wider spectrum of supporting tools to ensure maximum game-playing immersions.

There are several limitations of our study. The game elements studied and reported in this paper are mostly typical ones to support interaction and awareness. There are more elements designed to encourage player interactivity. In addition, our studies reported here only focus on the usability of awareness and interactions at a coarse level. A finer-grained level of usability analysis is desirable to answer questions such as, as a newly joined group member, can

I quickly and easily obtain information on the progress of other players? These are the focus of our future work.

REFERENCES

[1] W. Wright, "Models come alive (PC forum transcript)," http://many.corante.com/20030601.shtml.

[2] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)*, pp. 107–114, Toronto, Ontario, Canada, November 1992.

[3] C. Gutwin and S. Greenberg, "Design for individuals, design for groups: tradeoffs between power and workspace awareness," in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '98)*, pp. 207–216, Seattle, Wash, USA, November 1998.

[4] A. Lee, C. Danis, T. Miller, and Y. Jung, "Fostering social interaction in online spaces," in *Proceedings of the 8th IFIP Conference on Human-Computer Interaction (INTERACT '01)*, pp. 59–66, Tokyo, Japan, July 2001.

[5] N. Ducheneaut and R. J. Moore, "The social side of gaming: a study of interaction patterns in a massively multiplayer online game," in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '04)*, pp. 360–369, Chicago, Ill, USA, November 2004.

[6] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '04)*, pp. 72–81, Chicago, Ill, USA, November 2004.

[7] S. Smale and S. Greenberg, "Broadcasting information via display names in instant messaging," in *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '05)*, pp. 89–98, Sanibel Island, Fla, USA, November 2005.

[8] N. Ducheneaut, R. J. Moore, and E. Nickell, "Virtual "third places": a case study of sociability in massively multiplayer games," *Computer Supported Cooperative Work*, vol. 16, no. 1-2, pp. 129–166, 2007.

[9] N. Ducheneaut, N. Yee, E. Nickell, and R. J. Moore, ""Alone together?": exploring the social dynamics of massively multiplayer online games," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, vol. 1, pp. 407–416, Montreal, Quebec, Canada, April 2006.

[10] N. Ducheneaut and R. J. Moore, "More than just 'XP': learning social skills in multiplayer online games," *Interactive Technology and Smart Education*, vol. 2, no. 2, pp. 89–100, 2005.

[11] R. Oldenburg, *The Great Good Place*, Marlowe, New York, NY, USA, 1989.

[12] G. Simmel, *On Individuality and Social Forms*, University of Chicago Press, Chicago, Ill, USA, 1971.

[13] N. Nova, "Awareness tools: lessons from quake-like," in *Proceedings of Playing with the Future Conference*, Manchester, UK, April 2002.

[14] Baidu, http://post.baidu.com/f?kz=40033578, November 2007.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 321708, 6 pages
doi:10.1155/2008/321708

Research Article

Using a Camera Phone as a Mixed-Reality Laser Cannon

Fadi Chehimi, Paul Coulton, and Reuben Edwards

InfoLab21, Lancaster University, Lancaster LA1 4WA, UK

Correspondence should be addressed to Paul Coulton, p.coulton@lancaster.ac.uk

Received 29 September 2007; Accepted 31 January 2008

Recommended by Kok Wai Wong

Despite the ubiquity and rich features of current mobile phones, mobile games have failed to reach even the lowest estimates of expected revenues. This is unfortunate as mobile phones offer unique possibilities for creating games aimed at attracting demographics not currently catered for by the traditional console market. As a result, there has been a growing call for greater innovation within the mobile games industry and support for games outside the current console genres. In this paper, we present the design and implementation of a novel location-based game which allows us turn a camera phone into a mixed-reality laser cannon. The game uses specially designed coloured tags, which are worn by the players, and advanced colour tracking software running on a camera phone, to create a novel first person shoot-em-up (FPS) with innovative game interactions and play.

1. INTRODUCTION

Since the appearance of "snake" on the Nokia 16110 mobile phone in 1997, the potential for mobile games has excited the industry and market analysts alike producing forecasts of game revenues between 3.6 and 18.5 billion for 2006 [1]. However, the reality is that the mobile games have only generated approximately 2.6 billion in 2006, predominantly from games that were either cut down versions of old arcade games or very simple casual games [1]. Whilst these types of game no-doubt have their place in the market, there are increasing calls within this fledgling industry for developers to produce more innovative games and for operators to support and promote them on their portals.

This innovation will most likely be achieved by embracing both the challenges and opportunities [2] that mobile phones offer, particularly their mobility, connectivity, and rich feature sets which now include elements such as cameras and 3-D motion sensors. Mixed-reality games that enable users to see and interact with virtual computer generated content superimposed on views of the real (physical) world on mobile phones certainly would certainly fall into this category. This is because they generally take into account of the nature of the device often by incorporating location [3], proximity [4] and different modalities for user interaction [5] to create original game experiences.

Whilst location undoubtedly helps provide interactive game play, it is not always easy to achieve. It often requires additional hardware and/or software such as a global positioning system (GPS), or APIs to access Cell ID, or to utilise enhanced time difference of arrival (ETDOA) [3]. However, many of these features are uncommon in general handsets of these games to become successful, they would need to be able to be distributed. One way of ensuring this is to create games that run on as many handsets as possible and are not tied to a particular phone feature or fixed to a particular physical location. Therefore, part of the rationale of this research project is to produce a location-independent, mixed-reality game that utilises one of the most common features on mobile phone's camera. The camera has become almost a default feature on any mobile phone as evidenced by the fact that Nokia is looked at as the largest manufacturer of digital cameras in the world.

Whilst this is not the first mobile game to utilise the camera, it is unique when compared to previous games in that it uses advanced image processing of camera frames dynamically in real time. The captured frames are analysed on a frame-by-frame basis and fed to the application framework to provide the game interaction and status definition. The game could be regarded as a mobile mixed-reality version of paintball or laser tag, and hence it has been dubbed "Mobilazer."

To place the game in context with other available camera-based games, we will start with a brief review of the related games and research projects in this area.

2. OTHER PHONE CAMERA GAMES

Cameras are now a common feature of even the most basic mobile phone. Indeed, a reported 295.5 million were shipped in 2005 which represented nearly 40% of all phones shipped [6]. There is, thus, a real opportunity for their use within games but as very few innovative games have done so and in quite varied ways.

Some of the earliest such games, unsurprisingly, came out of Japan around 2003 such as "Photo Battler" from NEC which allowed players to turn photos into character cards that were then assigned various attributes enabling players to compete against each other. Around the same time "Shakariki Petto" appeared from Panasonic which was a virtual pet that a player fed by taking pictures of colours that represent food, for instance, the colour red represented apples. More recent games have also explored using the pictures themselves to create mixed-reality games such as the "Manhattan Story Mash-Up" [7] and "My Photos Are My Bullets" [8].

Other games have evolved to use the camera to detect movements of the phone and transfer them to movements within the game. Probably the best known are from game developer, Ojom, with its games "Attack of the Killer Virus" and "Mosquitos" for Symbian S60 mobile phones. In both games, the enemy characters that the player must "shoot" are superimposed on top of a live video stream from the mobile phone's camera. The player moves around this mixed-reality space by moving his phone and firing using the centre key of the joy pad. Other games have evolved to use visual codes to either detect movement [9], to imply, locate, or interact with objects such as ConQwest by Area Code [3].

However, none has yet utilised the complex real-time marker tracking, player recognition and interactivity identification in a real shoot-em-up game style as of that of "Mobilazer." There has been several marker tracking designs but almost all were based on black and white markers, whose recognition can be more efficient but has poorer information storage. In most cases, the tags were designed to: (1) get a piece of information instantly [10]; (2) feed-in input options to mobile phone applications [11]; (3) augment 3D models for interactive viewing [2, 5]; and (4) to locate landmarks for guiding visually impaired people [12].

3. GAME DESIGN

"Mobilazer" engages two or more teams of players in an unbounded physical location and employs camera phones to track special tags fitted on the "armour" worn by each individual player providing by this the means of game interaction. The specially designed software turns a player's mobile phone into a form of a "laser" cannon whereby he can shoot opposing players and interact with designated game objects such as team bases and poser-up collection points.

(a) Scanning for tag (b) Tag detected

Figure 1: Tag detection.

The armour-like vests worn by each player have coloured tags mounted on their front and back faces (although this could be extended to include smaller tags on arms and legs). The double-facing tags enable players to shoot others from a variety of orientations simply by pointing the phone's camera at the tag and aiming through the viewfinder (using the cross hair which is always in the middle of the screen) as shown in Figure 1(a). Once a tag is detected, "Mobilazer" displays a target sign centred at the coloured tag indicating the identification of the rival, and a box including his/her details (name, team, rank, weapons, etc.) as shown in Figure 1(b). The kills and points are controlled through interaction with a central game server through TCP/IP over the general packet radio service (GPRS).

3.1. *Game modes*

"Mobilazer" has been designed to accommodate four different playing modes which are selected by the preformed social grouping and controlled by a controller client unit (CCU) which is another mobile phone running as the game server.

3.1.1. *Fortress*

This game mode is a capture−the−flag style of game, engaging two or more teams. It starts by players representing each team agreeing on a location for their team base or Fortress which will be assigned a specific coloured tag. Each member of a particular team tries to protect his Fortress and attack the other teams' Fortresses and members, who are then effectively removed from the game once shot. The game continues until members of one group are the only survivors, or when one base is left intact while all others are destroyed.

3.1.2. *Last man standing*

This is a battle mode where a player tries to shoot all other players in the game to become the last man standing. This mode is highly flexible as it has no defined team bases to attack or defend. It can operate over a wide range of game arenas, and therefore the time taken to complete it is dependent

on the size of this arena. The game can be preset to terminate after a designated period of time where the winner will be the person with highest number of kills.

3.1.3. Individual battle mode

This mode is a timed free for all battle where no participant is locked out from the game session if shot. The aim is to score as many points as possible in order to acquire advanced equipment. Every time a player shoots an opponent, they scores points depending on the experience of the player being shot and the distance from which the shot was fired. Although players are not locked out of the game once shot, they have to wait a recovery period before their weapons are reactivated. This introduces a requirement to either hide or run.

3.1.4. Team battle mode

This battle mode is essentially a team version of the previous mode. Extra points are gained by each winning team member which he can use later to upgrade his/her weapons or armour.

3.2. Game balance

One of the essential elements for successful multiplayer computer games is differentiating players' experiences so that the game is balanced [13]. Balance means that all players feel they have an equal chance of competing, keeping in mind the differentiation between experienced and nonexperienced ones. Interestingly, this feature has not been given much consideration in mixed-reality games, although it likely ensures that players return to the game on multiple occasions. This is addressed in "Mobilazer" by introducing a variety of armours and weapons to allow both new players and more expert ones the opportunity to gain experience and be rewarded for repeated game play.

The system rates a player by the number of points they have collected. Certain points are gained depending on the status of the players they shoot and the distance between the two. These points qualify a player for a higher ranking and enable buying advanced gear each of which has a defined set of points to acquire and a number of hits to destroy as shown in Table 1.

Table 1: Mobilazer scoring scheme.

Equipment type	Points scored	Points needed	Number of hits
None	10	—	1
Bronze armour	20	30	2
Silver armour	40	60	5
Gold armour	60	90	10
Sniper	90	90	1
Tracker gun	100	200	1
Dist. ≤ 5 m	2	—	—
5 m < dist. ≤ 10 m	4	—	—
10 m < dist. ≤ 15 m	8	—	—
15 m < dist. < 20 m	12	—	—

4. GAME ARCHITECTURE

The Mobilazer architecture is composed of: the coloured tags, the mobile phone client application, and the mobile phone game server module as shown in Figure 2.

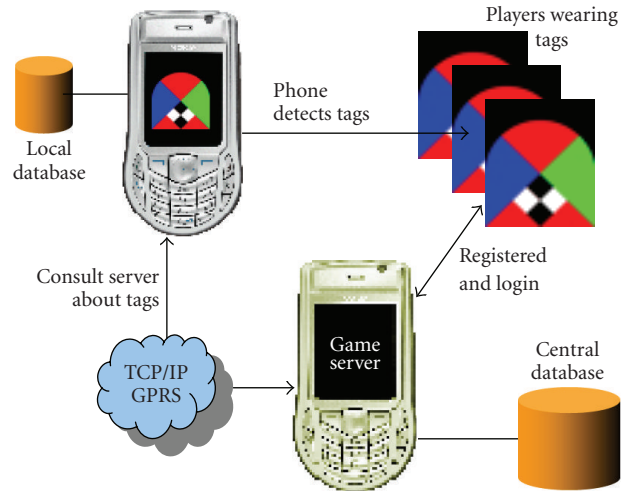


Figure 2: Mobilazer architecture.

4.1. Tag design

The coloured tag is the vital entity in the whole system as it identifies players in the game and initiates their interaction between players and the game assets. The tag shown in Figure 2 has been designed in this way to facilitate easy detection by cameras through the following four features:

(i) the triangular segments coloured red (top), green (right), and blue (left) which are used by the software to detect the centre point of the tag;

(ii) the upper black boundary facilitates calculation of the physical distance between players;

(iii) the black and white code area in the middle that contains the ID of the player or equipment;

(iv) the lower red two boundary triangles below the code which are used to improve the readability of the ID.

4.1.1. Physical structure

The tag has a large dimension of 20 cm by 24 cm to facilitate detection from longer distances and is made of "felt" fabric that reduces the amount of light reflected of its surface, which causes problems for the tracking software when reading the tag. This is indeed a known issue for phones using the camera in 1D and 2D barcode reading applications.

4.1.2. Triangular segments

The three coloured triangular segments ensure that even when the tag is rotated the vertical distance between the centre point and top of the segments is preserved, which is used in distance calculation. This holds true as long as the whole tag falls within the boundary of the captured camera frame and as long as it is upright and not rotated more than ±45 degrees from the upright.

4.1.3. Code blocks

The code area in the middle of the tag represents binary data and is capable of representing $2^4 = 16$ codes. The shape of a component block is not that important as tests have shown that it approximates to a square shape (pixels) when viewed from a distance. Rhombuses are used here just to optimise the space available and fit in as much blocks as possible while keeping them big enough to be identified.

4.1.4. Strategic points

There are three critical points in the design of the tag that ensure consistent recognition of the tag ID. The first point is the centre point where the tips of all coloured segments meet. It is necessary when calculating the distance between two players and identifying the upper corner of the code area. The second point is the right corner of the code area and the third is the left corner of the code. All three corners are used to read the ID through interpolation.

4.2. Client application

The client application runs on the phone and is responsible for

(i) communicating with the game server to inform it of shots, collected items, and synchronisation of players' information;

(ii) detecting tags, reading their embedded data, and displaying players' status and gear details in a box on screen.

When the Mobilazer client is launched, the player registers his/her nickname (used throughout the game), the team to which he/she belongs (if any), and the number associated with the tags on his/her armour. Each player submits these details to the server through connected TCP/IP sockets over GPRS and waits for acknowledgment to log in. When the client control unit (CCU) verifies that all players are registered it initiates the logging process.

Once in the game, the client detects a tag with a set of optimised image processing routines. It then reads the code on the tag and checks it against the records in its local database (which has been acquired from the server during the registration phase). Using a local database copy is more efficient since it decreases latency, reduces constant switching between the client and the server, and saves costs incurred from mobile network charges of GPRS use. The server will update this local database regularly once a player's status has changed by pushing the new data to all phones.

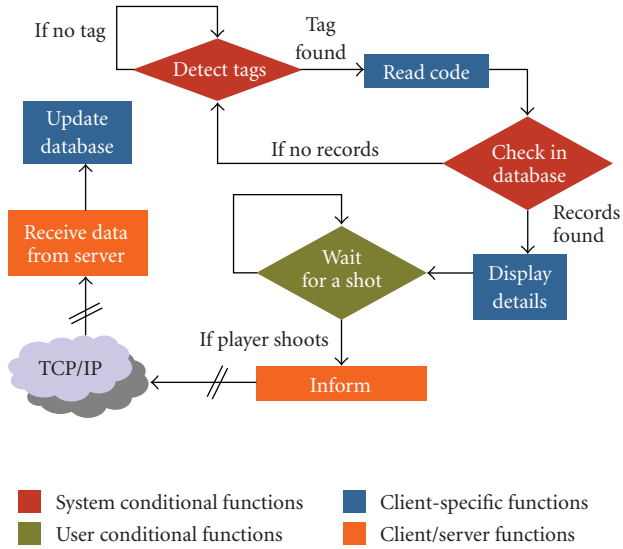


Figure 3: Client detection process.

When a tag ID is identified during the game, the application shows on phone screen a target sign centred on the tag and an information box containing the nickname of the target player, his/her team, his/her weapons, and his/her calculated distance from the shooter. If the attacker decides to shoot the opponent, the client sends a command to the server over TCP/IP through the connected socket requesting changing that opponent's status to indicate that he/she has been shot. The server then broadcasts this new information to all participants as well to update their local databases. This process is illustrated in Figure 3.

4.3. Game server

The server manages the communication and data exchange between players in a game session. Since the CCU operating the server will be present in the game field, it is used to define to the game server how many players will join a session before commencing the registration process. Then, the registration starts and the server displays on screen a counter of how many players have joined. When all players are registered, the CCU initiates the logging-in process. This two-phase initiation ensures that the server receives details of all players beforehand and then delivered all at once to each participant in the game.

The game server central database contains the nicknames of all players, their mobile phone IMEI numbers, their tag IDs, their groups (if available), their scores, and their acquired equipment. Players are identified uniquely by the IMEI numbers of their phones. Their weapons and scores will be associated with these numbers in the server database. Thus, each player must always use the same phone he used initially if he/she is to carry forward the gear possessed from previous battles to future ones.

5. GAME BALANCE FEATURES

To add depth to the game and, as we have highlighted previously, to aid the balance we have added four features that relate to players' experiences.

5.1. Distance measurement

The Mobilazer client is capable of measuring the distance between any arbitrary two players. The algorithm used to achieve this functionality is related to the tag. Once the centre point of a tag is found the application traverses the pixels above it vertically until it reaches a pixel on the arch between the red semitriangle and the black boundary. The number of pixels iterated r represents the radius length of the triangular. Experiment showed that a tag fills the screen area in Mobilazer when it is at 140 cm distance from the phone. In this case, the number of pixels between the centre point and the arch is 80 pixels. Based on these givens, the following formula gives us the new distance D (in centimetres and meters):

$$D_{\text{cm}} = Z \times \frac{80 \times 140}{r} = \frac{11200}{r} \Longrightarrow D_{\text{m}} = Z \times \frac{112}{r}, \quad (1)$$

where Z is the zooming factor defined in a subsequent paragraph.

5.2. Zooming functionality

The camera API in Symbian OS provides the functionality for camera zooming either digitally or optically. In our test device the Nokia 6630 has a digital zoom of up to 4X. The default zoom value of the API (1X) displays tags too small on screen to be recognised. So the application resets this value to 3.6X which allows players view objects on screen almost in their actual physical sizes.

To allow flexibility and enhance player experience we have added ten zooming options, and all players with all weapons except snipers have this functionality enabled.

Although calculating the distance depends on pixels and tag size on screen, the distance will still adhere to the exact physical distance even when the image is digitally zoomed in or out. This is achieved by preserving the proportions in (1) with the zoom factor Z. As 3.6X is the defaulted zoom value, Z is calculated as follows where z is the new zoom adjusted to the scene by the player. Note that the fraction in the denominator has been simplified to eliminate floating point division which optimises the system for mobile phone processors that are at present based on fixed-point routines:

$$Z = \frac{z}{3.6} = \frac{25 \times z}{45} = 0.534 \times z. \quad (2)$$

5.3. Sniper

The Sniper feature allows a player to shoot tags that are much farther than the normal camera range. The effect is applied using Symbian-specific bitmap manipulation utilities that enlarge and clip images, rather than simply depending on the camera API zooming functions. The image will be enlarged 6 times its size on screen, neglecting any zooming effect, and then clipped to the size of the mobile phone screen to produce a sniper close-up feel. Once this is done a black mask is superimposed on top of the view with sniper-detailed crosshair, as shown in Figure 4, to give the player the impression of looking through a telescopic sight. At start a sniper has 7 shots. The player can reload it either by collecting more points or by encountering an equipment collection point.

Figure 4: Sniper zoom.

5.4. Guided missile

This weapon is capable of locating its target even if it is not in line of sight. The user simply selects his/her enemy from the list of players downloaded from the server, and the guided missile finds its way to the unfortunate player. Since this is the most advanced piece of equipment, it requires the most number of points for a player to acquire and it has a limited number of shots. The process of loading this gun is similar to that of the sniper. Note that in case of "Fortress" mode the owner of a guided missile will not be able to target an opponent team's base remotely.

5.5. Armour shields

Table 1 highlighted the three different types of armours available in Mobilazer: bronze, silver, and gold. Beginner players will be provided the gold armours to provide the maximum protection as they learn how to play the game. As they become more experienced their armour level drops until they reach the point where they join a battle without any protection. However, it is still possible to acquire this protection during the game if a player collects enough points or triggers an equipment collection point.

Any armour can be combined with any other weapon. For example, a player may have a sniper gun and wears a silver armour. In this case the player will endure 5 hits instead of 1 before he is defeated.

5.6. Power-ups and gear collection points

Weapons, armours, and power-ups may be scattered around the battle field for players to collect and charge up. Special coloured tags can be registered in the server to represent such items. The players need to target their phone on the tag in the

same way as opponents and the corresponding item will be loaded to their account automatically once detected. Then, it will be updated in the server and forwarded through to other players. In case of picking up an armour by a player who already has one and has not been hit, the tag will have no effect, that is, the number of allowed hits will not be doubled. However, if the player has some hits, his/her armour will be renewed and hits will be eliminated. If a player picks up a weapon that he/she already has, his/her shots will be doubled.

6. CONCLUSION

This paper has highlighted the design and implementation of the game and as yet we have only completed controlled trials. In the next phase we intend to open up the game to a wide audience and present details of the user experience in a future publication.

With the current lack of innovation in the mobile games market, and in particular the failure to address much wider demographics means that new game genres should be considered for mobile gaming. Mixed-reality games create one such possible genre but it is often hampered by the fact that it demands utilising very high-end handsets with low critical mass. With this in mind we have shown that common camera phones have the potential for resolving this obstacle by creating highly sophisticated and immersive environments, and by they being capable of performing complex image processing tasks in real time.

ACKNOWLEDGMENT

The authors wish to acknowledge the support of Nokia for the real-time hardware and software laboratory in Infolab21 at Lancaster University where much of this work was carried out.

REFERENCES

[1] R. Tercek, "The first decade of mobile games," Keynote at GDC Mobile, San Francisco, USA, March 2007, www .roberttercek.com/.

[2] P. Coulton, O. Rashid, R. Edwards, and R. Thompson, "Creating entertainment applications for cellular phones," *ACM Computers in Entertainment*, vol. 3, no. 3, p. 3, 2005.

[3] O. Rashid, I. Mullins, P. Coulton, and R. Edwards, "Extending cyberspace: location based games using cellular phones," *ACM Computers in Entertainment*, vol. 4, no. 1, 2006.

[4] H. Clemson, P. Coulton, and R. Edwards, "A serendipitous mobile game," in *Proceedings of the 4th Annual International Conference in Computer Game Design and Technology (GDTW '06)*, pp. 130–134, Liverpool, UK, November 2006.

[5] P. Coulton, R. Edwards, W. Bamford, F. Chehimi, P. Gilbertson, and M. Rashid, "Mobile games: challenges and opportunities," in *Advances in Computers*, vol. 69, Elsevier Press, Amsterdam, The Netherlands, 2007.

[6] Nokia, "The mobile device market," August 2005, http:// www.nokia.com/.

[7] V. Tuulos, J. Scheible, and H. Nyhom, "Combining web, mobile phones and public displays in large-scale: manhattan story mashup," in *Proceedings of the 5th International Conference on Pervasive Computing*, vol. 4480, pp. 37–54, Toronto, Ontario, Canada, May 2007.

[8] R. Suomela and A. Koivisto, "My photos are my bullets—using camera as the primary means of player-to-player interaction in a mobile multiplayer game," in *Proceedings of the 5th International Conference on Entertainment Computing (ICEC '06)*, pp. 250–261, Cambridge, UK, September 2006.

[9] S. Bucolo, M. Billinghurst, and D. Sickinger, "User experiences with mobile phone camera game interfaces," in *Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia (MUM '05)*, vol. 154, pp. 87–94, Christchurch, New Zealand, December 2005.

[10] M. Rohs, "Visual code widgets for marker-based interaction," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '05)*, pp. 506–513, Columbus, Ohio, USA, June 2005.

[11] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan, "The smart phone: a ubiquitous input device," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 70–77, 2006.

[12] J. Coughlan, R. Manduchi, M. Mutuzaki, and H. Shen, "Rapid and robust algorithms for detecting colour targets," in *Proceedings of the 10th Congress of the International Colour Association (AIC '05)*, Granada, Spain, May 2005.

[13] C. Bateman and R. Boon, *21st Century Game Design*, Charles River Media, Rockland, Mass, USA, 2005.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 539078, 6 pages
doi:10.1155/2008/539078

Research Article

Using a Mobile Phone as a "Wii-like" Controller for Playing Games on a Large Public Display

Tamas Vajk,[1] Paul Coulton,[2] Will Bamford,[2] and Reuben Edwards[2]

[1] *Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Goldman Gyorgy ter 3. IV.em, H-1111 Budapest, Hungary*
[2] *Informatics, Infolab21, Lancaster University, Lancaster LA1 4WA, UK*

Correspondence should be addressed to Paul Coulton, p.coulton@lancaster.ac.uk

Received 26 September 2007; Accepted 12 November 2007

Recommended by Kok Wai Wong

Undoubtedly the biggest success amongst the recent games console releases has been the launch of the Nintendo Wii. This is arguably due to its most innovative attribute—the wireless controller or "Wiimote." The Wiimote can be used as a versatile game controller, able to detect motion and rotation in three dimensions which allows for very innovative game play. Prior to the Wii, and with much less furor, Nokia launched its 5500 model phone which contains 3D motion sensors. Using the Sensor API library available for the Symbian OS, this sensor data can be used by developers to create interesting new control schemes for mobile games. Whilst 3D motion can be utilized for ondevice games, in this paper we present a novel system that connects these phones to large public game screens via Bluetooth where it becomes a game controller for a multiplayer game. We illustrate the potential of this system through a multiplayer driving game using the Microsoft XNA framework and present preliminary feedback on the user experience from a public trial which highlights that these controls can be both intuitive and fun.

1. INTRODUCTION

Traditionally, the console game industry has divided potential players into two main categories: hardcore and casual gamers [1]. Of these categories it is the hardcore gamer that the industry has targeted itself towards as they exhibit features it wishes to exploit, such as [1] their tendency to purchase and play many games, their ability to enjoy longer play sessions, their ability to tolerate high levels of functionality in the user interface, their decision to play games as a lifestyle preference or priority. The result of the focus on this market has led to a console user demographic dominated by young white males and arguably the genres of first person shooter (FPS), sports and driving games [1].

However, there have been successful attempts to broaden the console user demographic, most notably by Nintendo through its DS console and innovative titles such as Nintendogs, WarioWare, and BrainAge, the latter of which created huge sales amongst previously nonconsole gamers.

The success of the DS led Nintendo to develop the Wii which has achieved phenomenal sales since its launch [2] by capturing the imagination of users not traditionally considered part of the current console gaming market [2]. From the perspective of both the game developer and game player, the most innovative feature of the Wii is the "Wiimote.'' The Wiimote itself was one of the primary design aspects of the Wii, and in an interview by Kenji Hall for Business Week in November 2006, Shigeru Miyamoto describes part of Nintendo's rationale:

"*The classic controller was something we had become fond of and gamers had become comfortable with. It had many important elements. But it also had come to dictate a lot of what went into games—the way graphics were made, the way battles were fought in role-playing games, the arc of in-game stories. They were all being made to fit one standard. Creativity was being stifled, and the range of games was narrowing.*"

Their solution was the Wiimote, a wireless controller which is able to sense both rotational orientation and translational acceleration along three-dimensional axes. It achieves this through the use of inbuilt accelerometers, together with a light sensor. This light sensor is used in conjunction with an array of light-emitting diodes centrally positioned above or below the console's display [3] which allows for six degrees of freedom. The Wiimote can be augmented with additional

features, one of which is the "Nunchuk," which features an accelerometer and a traditional analog joystick with two trigger buttons [3]. The overall result is a game interface capable of supporting a huge array of input possibilities that will enable new and exciting video game experiences.

Whilst mobile phones are in their relative infancy as gaming devices [4] compared to 7th generation consoles such as the Wii, their ubiquity and increasingly rich feature sets means they are equally capable of addressing the issue of widening the game player demographic [5]. Furthermore, as ubiquitous consumer devices they are ideal for interacting with urban computing environments and in particular large public displays [6]. Further, with the ever expanding feature set on handsets, mobile game developers can increasingly turn to innovative forms of user input such as RFID/NFC, cameras, microphone, and so forth [7].

Whilst some of the aforementioned input mechanisms have been the subject of a variety of research projects [7], the use of 2D accelerometers on mobile devices has been the subject of few studies [8] and the potential for using 3D accelerometers on phones appears completely unexplored [7]. This is principally due to the fact that the first 3D accelerometers have only just been integrated into mobile phones and these phones have yet to become widespread. The Nokia 5500 was one of the first such equipped phones having been targeted at sports users, utilizing the built-in motion sensors as a pedometer and speed/distance tracker for various exercising purposes. Other phones with motion sensors include Samsung's SCH-S310 (claimed to be the world's first 3D motion sensing mobile), NTT's N702, and more recently the Nokia N95.

Although there is an obvious potential for innovation in game play on mobile phones themselves, in this paper we are concerned with the potential use of a phone as a controller for games (or indeed a variety of applications) running on large public displays. In the following sections, we present the generic design and implementation of such an interactive system together with issues related to the implementation on this particular phone. Furthermore, we highlight the potential for the general use of accelerometers on phones in a variety of control interfaces. To illustrate the opportunities of this interface, we highlight its operation through the development of a novel multiplayer car racing game using an analogue control mechanism. This extends upon our previous work [9] based on the old arcade classic Tron Light Cycles in which a digital control scheme was employed. We then present the user experience from participants playing the game at a public event in Budapest, before drawing our overall conclusions.

2. PHONE CONTROLLER SYSTEM

Whilst there are obviously possibilities for creating innovative interfaces using 3D motion as the input mechanism for mobile games, in this project we explore using the phone as the games controller for multiplayer games shown on external displays and, in particular, large public screens. Using a large screen offers a number of benefits: it frees the games developer from constraints of the limited graphics capabilities of the mobile screen [10], enables a greater amount of movement to the participating players, provides a rich social atmosphere [6] and affords an opportunity for rich social interaction [11] in a variety of urban landscapes. However, we do acknowledge the practical deployment of such displays is often fraught with difficulty [12].

Figure 1: Poppet system.

The system developed is part of a generic framework, which we have termed Poppet,[1] for utilizing on-board phone sensors such as cameras, accelerometers, radio frequency identification/near field communications (RFID)/(NFC), which can be linked to games running on large public displays via Bluetooth as shown in Figure 1.

Although Poppet is capable of addressing a range of devices, in the next section, we specifically describe the design challenges involved in producing a mobile client for the Nokia 5500 together with its on-board accelerometers. However, the game server design is sufficiently generic to allow interaction with a wide range of phones using Bluetooth as the means of communication.

2.1. Mobile client design

Accessing the 3D motion sensors requires the use of the Symbian Sensor API which is similar in function to J2ME's Mobile Sensor API (JSR-256). Both of these APIs provide the potential to access a wide range of sensors such as accelerometers, thermometers, barometers, and humidity monitors, in fact any type of sensor designed to be incorporated in a mobile phone, or those accessible via Bluetooth [13]. Sensors need only be supported by the API library to be usable. The Symbian Sensor API is available from Nokia and requires the use of the Symbian S60 3rd Edition SDK. Whilst there is support for the Symbian Sensor API on several mobile devices, there are currently no mobile phones that include JSR-256 [13].

[1] In folk-magic or witchcraft, a "Poppet" is a doll made to represent a person, for casting spells on that person. The intention is that whatever are the performed actions, the doll is transferred to the subject. These dolls are often incorrectly referred to as Voodoo dolls.

Figure 2: Nokia 5500 accelerometer axes.

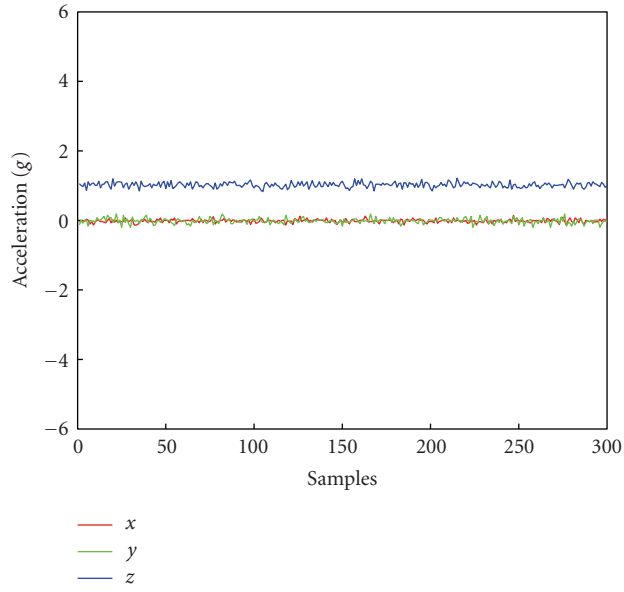


Figure 3: Nokia 5500 accelerometer data (phone at rest on a table).

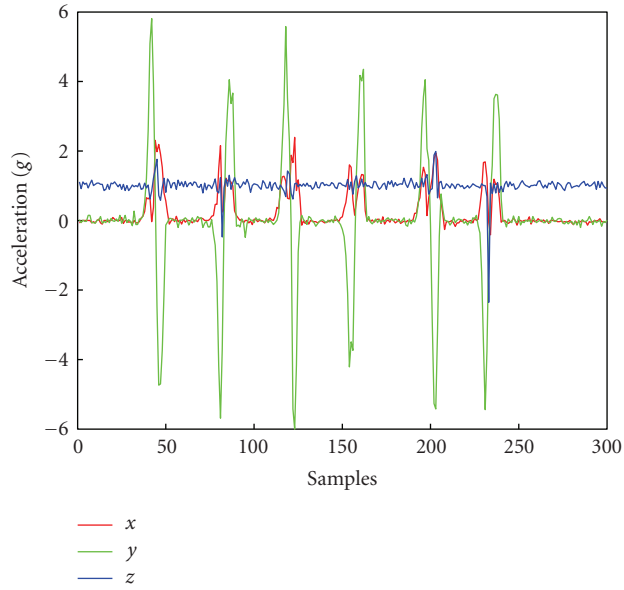


Figure 4: Nokia 5500 accelerometer data (lateral movement of phone across the table).

The general Poppet framework uses J2ME, as this is currently the most widely deployed mobile platform. Although the sensor data is not directly accessible from J2ME, because of the lack of JSR-256 as previously highlighted, the problem can be overcome by using a socket connection on the mobile phone to allow access to native services. The general solution for accessing native services from J2ME on Symbian S60 phones is by opening a low level socket connection in a Symbian C++ application then connecting to the defined port on the loop-back address from the J2ME application [14]. Thus the Symbian C++ application can retrieve sensor/other data from the phone and then forward this to any J2ME applications that may be listening. This solution has been applied in our simple mobile client to allow J2ME applications to access the 3D sensor data.

The connection between the game client and server is based on Bluetooth, which creates a reasonably high bandwidth (data rates can vary between 1 Mbps and a few Kbps depending upon the type of transfer mode initiated) between the devices. In order to allow a device to become discoverable by others, it is necessary to advertise at least one Bluetooth service. In the case of Poppet, the mobile client implementation uses the official Java Bluetooth API (JSR-82) to alleviate porting issues.

The Nokia 5500 utilizes a 6g accelerometer, that is, it can detect acceleration forces with a magnitude of up to six times that of earth's gravity. The accelerometer outputs three 12-bit signed data values at a frequency of around 37 Hz. These outputs correspond to the three phone axes (x, y, z) as shown in Figure 2.

In terms of illustrating the opportunities for creating novel interaction methods using phones with inbuilt accelerometers, it is worth considering the following three Figures (3, 4, 5) which show the recorded output from the sensors on the Nokia 5500 in three different scenarios. Note that output has been expressed as acceleration forces in g for clarity.

Figure 3 shows the three accelerometer outputs when the phone was statically placed upon a desk (representing a horizontal plane). It can be seen that outputs x and y are approximately zero (although some sensor noise is evident) and output z is showing a positive 1g force which is the effect of gravity on the device. Note, although sensor noise could be reduced by the application of a digital filter on the output values, in this case we felt showing the raw data was more informative. Overall, this figure illustrates that gravity provides a means of deducing the orientation of the phone which can then be utilized to provide a "tilt-based" controller.

In Figure 4, we see the accelerometer outputs resulting from several rapid lateral movements of the phone on a desk. At the start of the graph, we see outputs in the same state as in Figure 4 and then very large acceleration forces on axis y produced by the movement. Note that the output rapidly

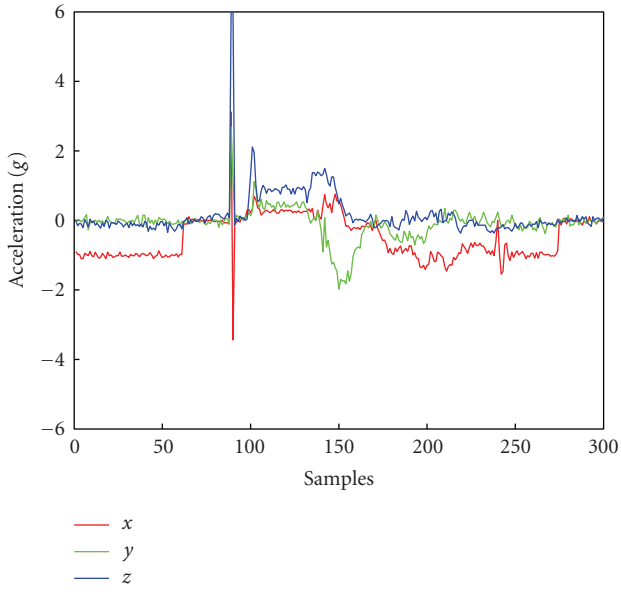


Figure 5: Nokia 5500 accelerometer data (phone dropped).

alternates between positive, then negative data values indicating the transitions from acceleration to deceleration and vice versa.

There are some smaller residual artifacts from this movement, seen in both x and z , which result from testing by hand rather than using a fixed jig (as it is difficult to isolate a movement completely under such conditions). Figure 4 depicts the potential for utilizing "hand gestures" for the control of events within games [15] and without the problems associated with using the camera to obtain the movement [15]. However, gesture recognition requires careful study as the variation in how the user holds the phone could produce anomalous outputs and there is no method of obtaining the phone's physical position within actual space as provided by the "sensor bar" for the Wii. This is because both rotation and translational acceleration affect the accelerometer's output (essentially they can produce the same internal forces on the accelerometer).

Figure 5 visualizes the output after the phone has been dropped. The trace shows that initially the phone is held upright with the screen held at the top with gravity acting on x. When the phone is dropped, the effect of gravity is overcome and all three accelerometers output values approximating zero, before the phone hits the floor with a jolt. The aim of this test was to show that the phone could be used to measure activity of a player within a game, such as jumping or running, whereby the phone would simply be worn rather than held and directly controlled.

2.2. Game server design

Using Bluetooth on a PC requires the implementation of a Bluetooth stack. Furthermore, different models of USB Bluetooth dongle have different stacks that may or may not be compatible with Microsoft Windows' Bluetooth stack. To alleviate some of these compatibility issues, the game server

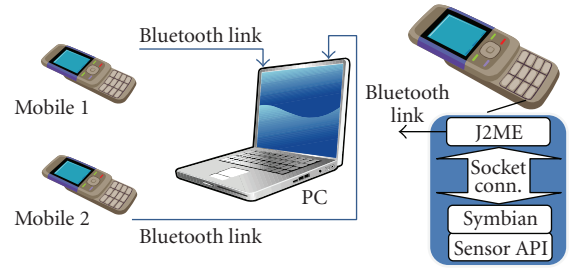


Figure 6: Phone game controller communication architecture.

was implemented using the Franson Bluetooth SDK for C#. This SDK supports the two most common Bluetooth stacks, the Windows stack and the WIDCOMM stack, which should ensure that the Poppet game server can be used with most Bluetooth devices. The basic communication architecture is shown in Figure 6 which also illustrates the on-phone socket communication used to transfer accelerometer data between the native application and a J2ME application.

The game server performs two main tasks: it locates/connects the Bluetooth phone controllers and it also processes the data to be used within the game. The first requires device discovery, which looks for the previously advertised service number on each phone controller in range of the PC. Once a phone controller is found, the PC tries to establish a connection with it. If connecting to the mobile is successful, both sides have all the necessary information to send and receive data between each other. The communication is based on streams at both ends.

The remaining task of the game server is to process the received data. As data triplets are arriving at the PC approximately 37 times per second which is fine in this case although, there might not be enough time to process every triplet and maintain the operation of the game in real-time. Therefore, if a real-time game is created, the developer has three options: they can drop packets, interpolate missing accelerometer samples, or assume that all samples can be processed before the next change in game state.

3. GAME IMPLEMENTATION

Even though the project presents some interesting technical challenges, it is principally aimed at developing a theoretical understanding of the user's experience in utilizing this type of tangible interface in conjunction with a game played on a public display. This understanding will be achieved by collecting empirical data from game prototypes played by various groups. Our first prototype was the game we called Mobi-Tron [9] which operated with a simple digital steering control and was tested amongst staff and students at Lancaster University.

When questioned on how to improve the experience, a number of players expressed a wish that the degree of tilt of the phone should correlate to the speed of the light cycle, as the current game only allows for changes in direction but not speed [9]. Other comments suggested incorporating sounds which the current version lacks and one person suggested

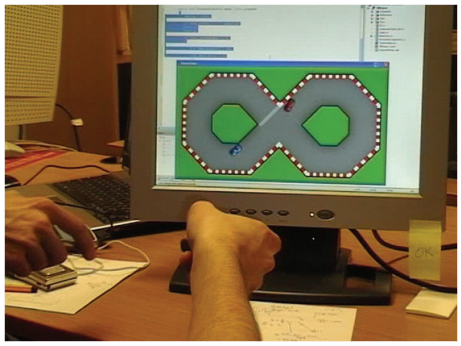


Figure 7: Screenshot of TiltRacer version 1.

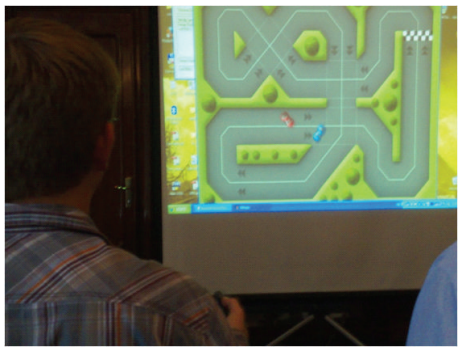


Figure 8: TiltRacer version 2 on a large public display.

that rather than simply turning at right angles, the full 360 degrees of movement should be used [9]. This has been subsequently incorporated in our new game called TiltRacer which is presented in this paper and shown in Figure 7.

TiltRacer utilizes the Poppet framework previously described and the actual driving game was developed in C# using Microsoft's XNA game framework. The initial version of the game was a simple figure of eight shaped track shown in the previous figure, although this was expanded to a more complex track, shown in Figure 8, for the user trial as our initial in-house feedback considered it too easy to master. The game itself is a simple first to complete a selected number of tracks and can be played by up to 4 players simultaneously although in the trials we limited this to two.

4. USER EXPERIENCE

In this section, we present the results of testing the system amongst participants of the Forum Nokia Technical Days and European Mobile Innovation Competition[2] held between the 6th and 8th June 2007 in Budapest, Hungary. The game was displayed on a large screen at the evening reception for the event is shown in Figure 8 and participants were invited to try the game out. To ascertain aspects of the user experience

[2] The game represented the UK entry for the innovation completion, having previously won through the UK heat, and was the ultimate winner of the event.

Figure 9: Users playing TiltRacer.

as a whole we decided to adopt and ethnographic approach [16] because of the nature of the event which combined video and still image recording by three researchers of the 30–35 individuals who tried the game.

The general response to the interface was that it was fun, as highlighted by some of the players' facial reactions in Figure 9 and that it was also intuitive, as participants quickly gained an understanding of how their movements affected game avatars with little or no instruction from the research team or preceding players. Further, as is evidenced by Figure 9, the players were concentrating solely on the screen and not looking at the interface which aided their immersion in the experience. Whilst this observation is interesting in relation to research concerning the display of public information on a large screen against private information on a small screen it should be noted that this largely influenced by the nature of the game-play and something like a card game would provide more interesting analysis of this area.

The overall response to the game is probably best summed up by Harri Lehmuskallio, a user experience specialist for Idean who was part of the judging panel,

"*The application expands the interaction from mobile to the physical world. The game concept is simple, but it demonstrates that casual games can be played this way in public spaces. The fact, that TiltRacer was created, opens new possibilities for the industry as it shows that innovating is fun and doable.*"

5. CONCLUSIONS

The Wii console has captured the imagination of users who previously have not participated in console-based gaming. Whilst the particular game genres chosen are a factor in user acceptance, it is undoubtedly its innovative controller functionality that has proved its most attractive feature.

Whilst we are certainly not suggesting that mobile phones could rival the Wii, the inclusion of 3D accelerometers opens up the possibility for the same type of innovation for games running on mobile phones. Further, they also provide users with an innovative game interface that they already carry around as part of their everyday day lives. This could allow them to interact in novel ways with games forming part of the environment. In particular, the increasing presence in our cities of large public displays is making this hybridization of

virtual and real space available to the mass market and thus brings new opportunities for games.

Although this game can be considered a fairly simple example in terms of graphics and complexity, and there is certainly a requirement for further study using other types of game, it does highlight that a novel interaction mechanism coupled with a fun group activity can provide an enjoyable social experience, with high levels of user interaction.

ACKNOWLEDGMENT

The authors thank Nokia for the provision of software and hardware to the Mobile Radicals research group which were used in the implementation of this project.

REFERENCES

[1] C. Bateman and R. Boon, *21st Century Game Design*, Charles River Media, Rockland, Mass, USA, 2005.

[2] J. Brightman, "Merrill Lynch: 30% of U.S. Households to Own Wii by 2011," http://biz.gamedaily.com/industry/feature/?id=15309.

[3] Nintendo, "Nintendo Nsider Technical Forums," http://forums.nintendo.com/nintendo/.

[4] R. Tercek, "The First Decade of Mobile Games," Keynote at GDC Mobile, San Francisco, Calif, USA, March 2005, www.roberttercek.com.

[5] P. Coulton, R. Edwards, W. Bamford, F. Chehimi, P. Gilbertson, and M. Rashid, *Mobile Games: Challenges and Opportunities, Advances in Computers*, Elsevier Press, Amsterdam, The Netherlands, 2007.

[6] J. Leikas, H. Stromberg, V. Ikonen, R. Suomela, and J. Heinila, "Multi-user mobile applications and a public display: novel ways for social interaction," in *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '06)*, pp. 66–70, Pisa, Italy, March 2006.

[7] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan, "The smart phone: a ubiquitous input device," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 70–71, 2006.

[8] J. F. Bartlett, "Rock 'n' scroll is here to stay," *IEEE Computer Graphics and Applications*, vol. 20, no. 3, pp. 40–45, 2000.

[9] T. Vajk, W. Bamford, P. Coulton, and R. Edwards, "Using a mobile phone as a 'Wii like' controller," in *Proceedings of the 3rd International Conference on Games Research and Development*, Manchester, UK, September 2007.

[10] P. Coulton, O. Rashid, R. Edwards, and R. Thompson, "Creating entertainment applications for cellular phones," *ACM Computers in Entertainment*, vol. 3, no. 3, 2005.

[11] J. Reid, J. Hyams, K. Shaw, and M. Lipson, "Fancy a Schmink?: a novel networked game in a café," *ACM Computers in Entertainment*, vol. 2, no. 3, p. 11, 2004.

[12] O. Storz, A. Friday, N. Davies, J. Finney, C. Sas, and J. G. Sheridan, "Public ubiquitous computing systems: lessons from the e-campus display deployments," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 40–47, 2006.

[13] P. Coulton, W. Bamford, F. Chehimi, P. Gilberstson, and O. Rashid, "Using in-built RFID/NFC, cameras, and 3D accelerometers as mobile phone sensors," in *Mobile Phone Programming: Application to Wireless Networking*, F. H. P. Fitzek and F. Reichert, Eds., pp. 381–396, Springer, New York, NY, USA, July 2007.

[14] A. Gupta and M. de Jode, "Extending the reach of MIDlets: how MIDlets can access native services," Symbian Technical Paper, version 1.1, June 2005.

[15] S. P. Walz, R. Ballagas, J. Borchers, et al., "Cell spell-casting: designing a locative and gesture recognition multiplayer smartphone game for tourists," in *Proceedings of PerGames*, pp. 149–156, Dublin, Ireland, May 2006.

[16] A. Crabtree, S. Benford, C. Greenhalgh, P. Tennent, M. Chalmers, and B. Brown, "Supporting ethnographic studies of ubiquitous computing in the wild," in *Proceedings of the Conference on Designing Interactive Systems (DIS '06)*, vol. 2006, pp. 60–69, University Park, Pa, USA, June 2006.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 378485, 7 pages
doi:10.1155/2008/378485

Research Article

Game Portability Using a Service-Oriented Approach

Ahmed BinSubaih and Steve Maddock

Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK

Correspondence should be addressed to Steve Maddock, s.maddock@dcs.shef.ac.uk

Received 30 September 2007; Accepted 7 January 2008

Recommended by Wong

Game assets are portable between games. The games themselves are, however, dependent on the game engine they were developed on. Middleware has attempted to address this by, for instance, separating out the AI from the core game engine. Our work takes this further by separating the *game* from the game engine, and making it portable between game engines. The game elements that we make portable are the game logic, the object model, and the game state, which represent the game's brain, and which we collectively refer to as the game factor, or G-factor. We achieve this using an architecture based around a service-oriented approach. We present an overview of this architecture and its use in developing games. The evaluation demonstrates that the architecture does not affect performance unduly, adds little development overhead, is scaleable, and supports modifiability.

1. INTRODUCTION

The shift in game development from developing games from scratch to using game engines was first introduced by Quake and marked the advent of the game-independent game engine development approach [1]. In this approach, the game engine became *the collection of modules of simulation code that do not directly specify the game's behaviour (game logic) or game's environment (level data)* [2]. The game engine is thus reusable for (or portable to) different game projects. However this shift produces a game that is dependent on the game engine. For example, why can't a player take his favourite game (say Unreal) and play it on Quake engine or Quake game on Unreal engine?

Hardware and software abstractions have facilitated the ability to play a game on different hardwares and on different operating systems. These abstractions have also facilitated the ability to use data assets such as 3D models, sound, music, and texture across different game engines. This ability should also be extended to allow for the game itself to be portable. The goal of our work is to make the game engine's brain portable, where the brain holds the game state and the object model and uses the game logic to control the game. We collectively refer to these three things as the G-factor.

We see the portability of the G-factor as the next logical step in the evolution of game development and, following Lewis and Jacobson's terminology [1], we call it the game-engines-independent game development approach. A benefit of making the G-factor portable would be to encourage more developers to make use of game engines, since a particular game engine's future capability (or potential discontinuation, as was the fate of Adobe Atmosphere which was used for Adolescent Therapy-Personal Investigator [3]) would not be a worry as a different game engine could easily be substituted. This problem has recently been referred to as *the RenderWare Problem* [4] after the acquisition of RenderWare engine by Electronic Arts (EA) and its removal from the market. We see the issue of rewriting the G-factor from scratch every time we migrate from one engine to another as similar to the undesired practice of developing games from scratch which was deemed unfeasible and resulted in the advent of game engines.

As we noted earlier, portability is an issue that pervades all games with regards to game assets. In addition, however, and related to our work, are the moves towards addressing more aspects of portability. Examples include artificial intelligence (AI) architectures and interfaces [5]. AI architectures use custom made or off-the-shelf components such as AI Middleware (e.g., SOAR [6] or AI.Implant (http://www.biographictech.com (accessed 5/5/2007)). However, specifying the game using the AI middleware format merely moves the game from one

Table 1: Comparing a typical game development approach to GSA's approach

Step	Typical approach	GSA's approach
(1) Create the level data.	Create the decorative objects in the game engine.	
	(i) Create the game objects using the world builder or TorqueScript.	(i) Create the game objects using the world builder in the game engine and give them a unique ID which identifies these objects in the game space as well. Load these objects using TorqueScript. (ii) Create the game objects in the game space with the same unique ID using Jython.
(2) Create the GUI.	Use the game engine interface builder or TorqueScript to create the interface. The behaviour is set as part of the game logic (step 4).	
(3) Create the object model.	(i) Use TorqueScript to extend the objects or create new ones.	(i) Create the object models for the game objects that require representation in the game engine and the game space. (ii) Create the other game object models in game space.
(4) Create the game logic.	(i) Use TorqueScript to set the behaviour in the game engine.	(i) Use Jython or Java to create the logic in the game space.
(5) Create the adapter.		(i) Send the updates from the game engine to the game space. (ii) Create the adapter which translates between the game engine and the game space.

proprietary format (game engines) to another (AI middleware). The work on interfaces aims to facilitate access to game engines. For example, Gamebots [7] and GOLOG Bots [8] are the interfaces that have been used to access Unreal, with, similarly, Quakebot [9] for Quake, FlexBot [10] for Half-Life, and Shadow Door [11] for Neverwinter Nights. These provide interfaces for specific game engines. Other projects are attempting to provide common interfaces to game engines such as the initiative by International Game Developers Association (IGDA) for world interfacing [12] and OASIS [13]. Despite this work, such interfaces may have more success in the serious games community rather than the fast-evolving games industry.

In [14], we described, in detail, how to make the G-factor portable. In this paper, we give an overview of this earlier work, and instead focus more on the evaluation process, addressing issues such as performance, implementation overhead, scalability, and modifiability. We present results of conducting both an unstructured evaluation process and a structured evaluation using ATAM [15], and contrast the two in the subsequent discussion.

The remainder of this paper is structured as follows. Section 2 demonstrates the issues with the typical game development approach through the development of a sample game. This is then contrasted with the development of the same game using our approach, which enables the G-factor to be portable. Section 3 describes the evaluation process and what it revealed about the two development approaches. Finally Section 4 presents the conclusions.

2. AN ARCHITECTURE FOR G-FACTOR PORTABILITY

This section contrasts a typical game development approach with the game-development approach proposed in our work. Section 2.1 describes what is considered to be a typical development approach through the development of a sample game, and highlights the dependencies associated with this approach. Section 2.2 then proposes an approach to address these dependencies and describes an architecture called game space architecture (GSA) which has been implemented to validate this approach.

2.1. A typical approach to game development

We will use a game that we call *Moody NPCs* to illustrate the typical approach to game development. The game consists of a number of nonplayer characters (NPCs) that react to a player based on their mood. The player can carry out actions such as greeting or swearing. Each NPC reacts to the action based on his mood which is governed by two variables: cowardness/courage and forgiveness/punishment. The game allows the user to navigate the level and click on an NPC which reveals its current mood and the actions available. The player can adjust the mood variables and try out different actions. The Torque game engine is used to demonstrate how the game is developed.

The typical game development approach can be grouped into four main steps as shown in the typical approach column in Table 1. To create the game level data (step 1), Torque engine provides a level editor called World Editor. The level can also be created using other ways such as: scripting, API, and configuration files. The game level data contain the terrain of the environment and the decorative objects (e.g., houses, trees, etc.). The level also contains location markers for the game objects (e.g., NPCs and player). Scripting is used to create the other game objects (e.g., reaction, action, and interaction). This approach for creating the game level data is very common amongst game engines—84% of engines we surveyed provided editors to create the game level [5].

Figure 1 shows the graphical user interface created in step 2. This has mood variable sliders on the top left corner of the screen and an actions controller on the bottom left corner of the screen. The player can use the keyboard to navigate

Figure 1: The Moody NPCs game.

around and the mouse to select an NPC. We used Torque's GUI Editor to set the interface controllers, although it is also possible to use scripting and configuration files.

Step 3 is to create the object model to hold the structure for the game objects. The object model consists of five classes: player, NPC, action, reaction, and interaction. Torque has a default object model for the player and the AI player. We extended these to add the properties that are specific to the game (i.e., mood variables for an NPC). We created the other classes using a static object model using TorqueScript. The other game object models are created using scripting. Finally, step 4 is to create the game logic which controls how the NPC reacts to the player actions.

2.2. GSA's approach

Figure 2 illustrates the software dependencies problem GSA is aiming to tackle. the example used is the development of *Gears of War* , which is dependent on Unreal Engine 3 and the underlying software [16]. This is similar to the dependency the Moody NPCs game suffers from, and also to the dependencies exhibited by the projects we surveyed in an earlier paper [5].

GSA's objective is to reduce the dependencies by adopting a service-oriented design philosophy, which enables the G-factor to exist independently of the game engine. The service-oriented approach has proved its practicality for achieving different types of portability such as platforms and languages [17]. The novel design approach employed in GSA combines a variant of the model-view-controller (MVC) pattern to separate the G-factor (i.e., model) from the game engine (i.e., view) with on-the-fly scripting to enable communication through an adapter (i.e., controller). The use of a variant of MVC rather than the normal MVC avoids a known liability where the view is tightly coupled to the model [18]. The use of on-the-fly scripting is used to maintain the attractive attributes associated with typical game development where data-driven mechanisms are used to modify the G-factor. Most notably, modifiability is upheld in the typical game development approach using scripting, which our surveys found to be very popular with game engines and projects that use game engines [5]. To maintain this level of modifiability (i.e., scripting level access) to the game engine and the game space, GSA uses on-the-fly scripting to communicate with both via the adapter. For example, a communication may begin with the game engine sending the updates to the adapter (step 1 in the communication protocol shown in Figure 3). The adapter converts them into scripts or direct API calls (step 2) which are then used to update the game space (step 3). When the game space needs to communicate with the game engine, it notifies the adapter of the changes that need to be communicated (step 4). The adapter formats these into the engine's scripting language (step 5) and sends them to the engine to be executed (step 6). The separation and the communication mechanisms allow the G-factor to exist independently of the game engine. The effect this has on portability is that when migrating to a new engine, the elements in the game space (i.e., the game state, object model, and game logic) can stay intact. Contrasting this to migrating a game developed using the typical game development approach, which often require all three elements to be created again, shows the extent of the effort saved.

As was shown in Table 1, the first difference between our approach and the typical game development approach is the creation of the game objects, which is split over the game engine and the game space due to the two types of game objects. The first type is the game objects that have to have representations inside the game engine to provide visual representations, such as the player and the NPCs needed for the Moody NPCs game. These require real-time processing in the game engine, and it is impractical to communicate every frame from the game space to the game engine. Therefore, these objects have to be created in the game engine as well as the game space and only updates are communicated. The second type of game objects is the ones that do not have representations inside the game engine, such as the action, interaction, and reaction objects. These objects can be created in the game space only. The object model creation is similarly split over the game engine and the game space. The second difference is creating the game logic in the game space rather than the game engine. The third difference is creating the adapter which handles the communication between the game space and the game engine.

3. EVALUATION AND DISCUSSION

A software architecture can be evaluated using structured or unstructured methods. An unstructured evaluation, which is a common way to evaluate a software architecture [19], consists of randomly throwing challenges at the architecture and hoping that either the architecture can address them, or that they will reveal its limitations. In structured evaluation, methods such as ATAM [15], SAAM [20], ARID [21], and ABAS, PASA and CBAM [22] are used to probe the architecture with the aim of exercising the whole architecture. We used ATAM in our structured evaluation, a method that is not limited to a particular stage of the development cycle, and which involves stakeholders (i.e., user, maintainer, developer, manager, tester, architect, security expert, etc.) in specifying the architecture attributes to address. In the following paragraphs, we will summarise the findings of detailed structured [23] and unstructured [24] evaluations carried out in our earlier research papers. We will focus on

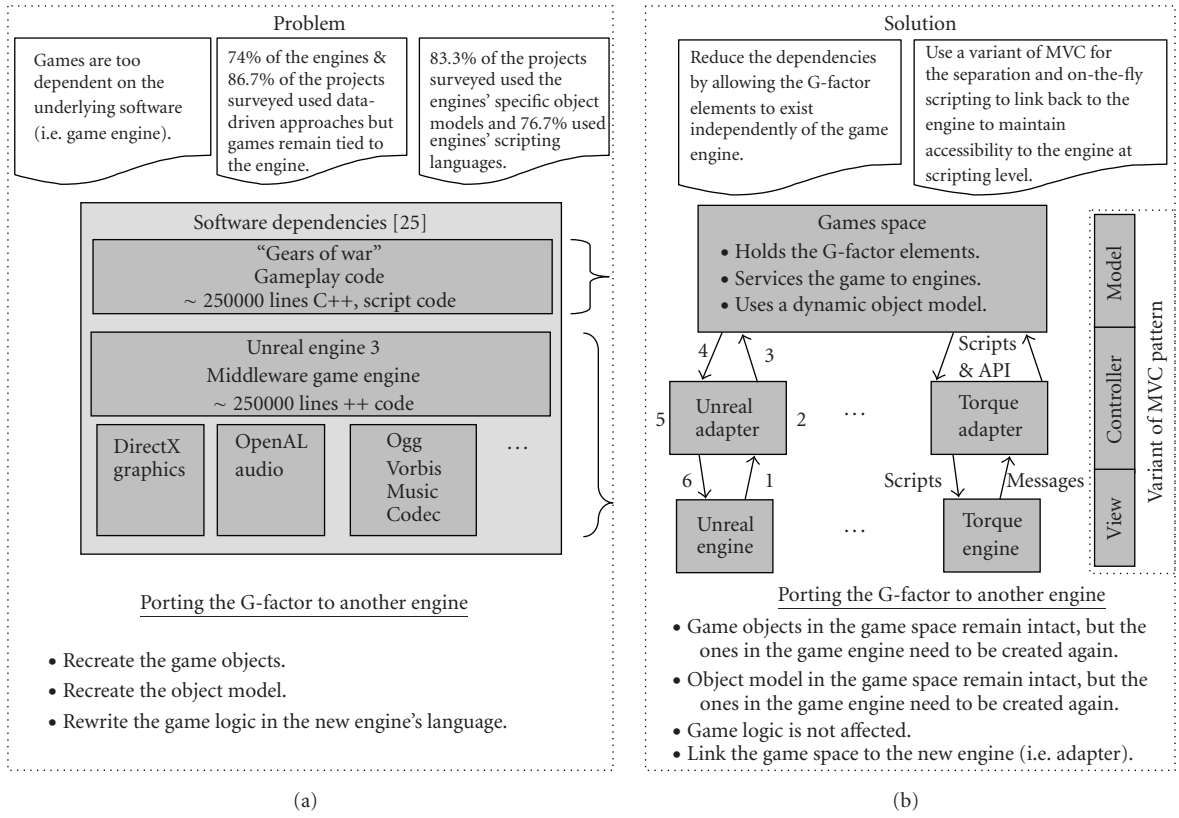


Figure 2: GSA overview. The numbers highlighting the communication between the game space and the game engine are described in Figure 3.

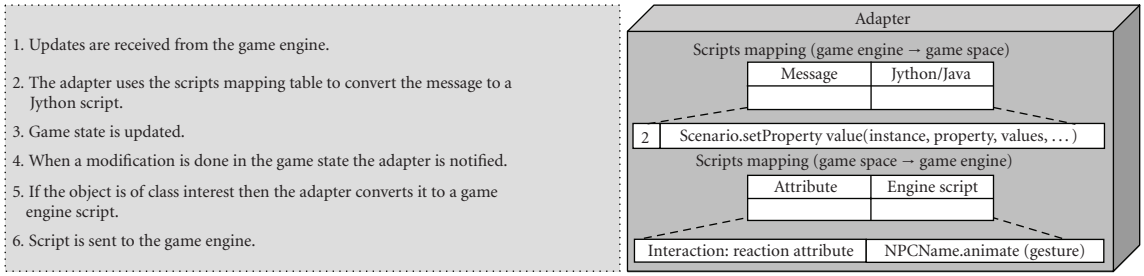


Figure 3: Communication between the game engine and the game space.

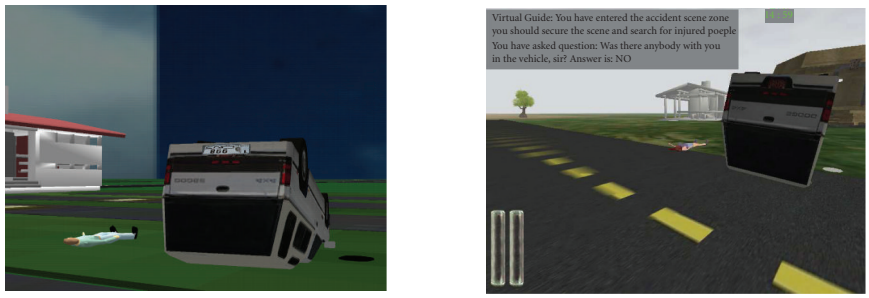
four attributes: portability, performance, modifiability, and scalability. Following this, we contrast the structured and unstructured approaches to evaluation.

3.1. Portability

The unstructured evaluation found that GSA managed to address the portability challenge by servicing the same G-factor to two different engines [13]: a bespoke engine developed on top of DirectX 9.0 and the Torque game engine (see Figure 4). This was done without modifying the G-factor and was constrained to modifying the adapter. Similarly, the structured evaluation found GSA supports portability. It found that the separation using the MVC pattern allows for better portability since it allows for multiple views (i.e., game engines) for the same model (i.e., G-factor). In addition, the structured evaluation found that portability could be undermined if the game engine does not fully expose the required functionality through scripting since the adapter relies on scripting for communicating back to the game engine (see Figure 2).

3.2. Performance

The aim here was to find the average reduction in frames-per-second (fps) due to the use of GSA. To get a performance indicator, a player was simulated to be running continuously around a path for 30 minutes (see Figure 5). Using this simulation, two performance tests were run to contrast the overheads of a game developed with the typical development



(a) (b)

Figure 4: (a) Smart terrain running on bespoke engine, (b) the same G-factor running on Torque [24].

approach to one developed using GSA. The performance overheads measured were: fps, CPU, memory, and network (for the test using the game space). The average reduction in fps was 11.69% when following the GSA approach. This average fps reduction is relatively large for a small game and more tests need to be performed to get a better indication of how this reduction will scale with the game size. However, when comparing this finding to the findings from the scalability challenge (described later), we find that GSA does not affect performance unduly. The structured evaluation revealed two issues. It found that the data integrity across the different game states (i.e., game engine and game space) was at risk. This is due to the delays that might occur because of the separation as a result of the use of the MVC pattern which add an overhead for exchanging information. Initial tests revealed no problems, but further tests are required before this can be established with certainty. In addition, there is a danger if the message load increases that the game space becomes the bottleneck in the architecture.

3.3. Modifiability

Here, the success of GSA was judged by the ability to create different G-factors on the same architecture using a different object model and game logic. The fact that different G-factors (Figures 1, 4, 5, and 6) can be developed using GSA showed its modifiability. In addition, a structured evaluation process measured the modifiability across the different parts of GSA by examining how each architectural decision affects modifiability, and how it trades against the other quality attributes (e.g., portability and performance) [23]. The evaluation revealed that if a single unique identifier cannot be set for game objects on the game space and game engine, then GSA becomes very sensitive to any modification as it has to be added manually in the adapter. Furthermore, using on-the-fly scripting allows for better modifiability but runs slower than precompiled code. Modifiability is also enhanced by the use of a variant of the MVC pattern that reduces the dependencies between the model and the view.

3.4. Scalability

The aim was to identify how much overhead is added as the game size grows. This was examined by developing a serious

Figure 5: First-person shooter game [24].

Figure 6: A serious game for traffic accident investigators [25].

game for traffic accident investigators [25] (see Figure 6). The adapter's implementation overhead for each challenge is presented in Table 2. Using the implementation overheads of the adapters compared to their game logic sizes in each of the test games developed, we can forecast that for small game size, the overhead is large, but that it stabilises at around 6% for code of size between 100,000 and 500,000 lines (see Figure 7) (http://support.microsoft.com/kb/828236 (accessed 24/8/2007)). The scalability challenge also showed that the performance overhead was not noticeable when judging its success in training [25] for which smooth play is crucial to avoid frustrating the users. The structured evaluation found that using a dynamic object model allows for better game-model scalability, but it makes the architecture very sensitive to change as the change propagates to the game logic and to the adapter.

Table 2: The implementation overhead for the adapter.

Challenge	Logic size (lines of code)	Adapter size (lines of code)
Portability	60	346 (bespoke) 354 (Torque)
Modifiability	100	350
Performance	70	300
Scalability	6214	1100

3.5. Structured versus unstructured evaluation

The unstructured evaluation revealed how well the architecture can cope with the challenges. However, there was no easy way to establish the correlation between the challenge and what architectural decisions had supported or undermined. Furthermore the unsystematic way of generating scenarios (i.e., challenges) meant that some time was unnecessarily spent in implementing different tests when one could have served all the challenges (e.g., the implementation of the serious game (see Figure 6) used in the scalability challenge could have been used to test all of the challenges). This could be attributed to the incomplete overall evaluation picture due to the lack of systematic guidance. Although, there is no guarantee that a structured evaluation would not produce redundant probing since, just like the unstructured evaluation, it is also scenario-based. However, the chances are reduced due to the fact that the generation of scenarios is guided by using a utility tree (The utility tree elicits the quality attributes down to the scenario level to provide a mechanism for translating architectural requirements into concrete practical scenarios) in which all the scenarios are identified. This serves two purposes. The first purpose is that once all the scenarios are present, the experimentation can begin by choosing a test where preferably all these scenarios can be addressed. The second purpose is that it describes the decisions that are going to be analyzed by the scenario which means that any repetitive probing can be identified.

The problem with scenario-based evaluation which both unstructured and structured evaluations use is that the evaluation is only as good as the scenarios generated, which in turn depends on the stakeholders in the evaluation team. Although there are measures put in place to ensure that the selection includes all the important personnel (i.e., architects and domain experts), the fundamental problem still persists.

Contrasting ATAM's output to the unstructured evaluation results, which quite often answer the challenge with yes or no, or with some metrics such as network load or fps, highlights the strengths of ATAM. ATAM classifies the decisions according to how they affect the architecture (i.e., support or undermine it). We found the ATAM process helpful in understanding our architecture better. Of further benefit is that it should also act as a guide when there is a need to modify or evolve GSA. This guidance is based on the fact that it reveals the strengths and weaknesses of the architectural decisions. In future, we recommend using ATAM alongside the development cycle. This is where ATAM is designed to be most effective by revealing issues at different stages of the development cycle when they are cheaper to address. Had we started with ATAM, we believe it would have saved us time and effort by avoiding the creation of a number of redundant challenges.

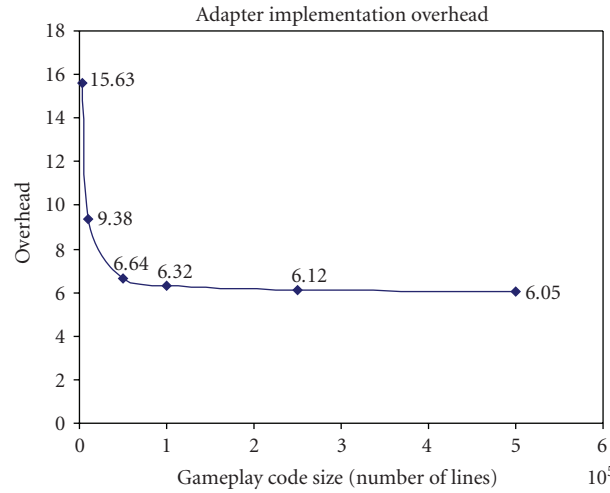


Figure 7: The adapter code forecast compared to the game logic code size.

4. CONCLUSIONS

We have presented an architecture for making *games* (i.e., G-factors) portable between game engines. The changes required to the typical game development approach have been demonstrated through the development of a sample game called Moody NPCs. In addition, the work has presented the findings from two types of evaluation. The findings have revealed that GSA is capable of making the G-factor portable, but GSA adds performance and implementation overheads. Despite these overheads, GSA has been shown to scale to real world applications [25]. Modifiability has been found to be sensitive in cases where a unique identifier cannot be set for game objects.

Whilst the unstructured evaluation managed to reveal issues with the architecture, the mechanism of throwing random challenges resulted in redundant challenges and failed to articulate which architectural decisions undermined or supported GSA. Using ATAM guided the evaluation better. Employed earlier, it could have helped to avoid the redundancy in the unstructured evaluation. Also, it was capable of revealing how the architectural decisions interact in order to support the required attributes. Although the portability presented in this work has only been shown across two engines, the approach followed to achieve that is consistent in the way the two engines were linked via the adapter, and, therefore, there is no reason why it cannot be followed to link other engines.

With gameplay predicted to be the distinguishing factor between future games [26] and combined with the increased number of commercial licensees of game engines and the increased interest from the serious games community, this will increase the need for portable games for two reasons. The first reason is because developers can keep the visual aspects of their game up-to-date with the latest game engine.

The second reason is the security from having to face *the RenderWare Problem* [4]. However, the incentive for game engine developers is less clear.

REFERENCES

[1] M. Lewis and J. Jacobson, "Games engines in scientific research," *Communications of the ACM*, vol. 45, no. 1, pp. 27–31, 2002.

[2] J. Wang, M. Lewis, and J. Gennari, "Emerging areas: urban operations and UCAVs: a game engine based simulation of the NIST urban search and rescue arenas," in *Proceedings of the 35th Winter Simulation Conference*, pp. 1039–1045, New Orleans, La, USA, December 2003.

[3] D. Coyle and M. Matthews, "Personal investigator: a therapeutic 3D game for teenagers," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '04)*, Vienna, Austria, April 2004.

[4] S. Carless, "Rise of the game engine," *Game Developer*, pp. 2–2, 2007.

[5] A. BinSubaih, S. Maddock, and D. Romano, "A survey of 'game' portability," Tech. Rep. CS-07-05, Department of Computer Science, University of Sheffield, Sheffield, UK, 2007.

[6] J. E. Laird, M. Assanie, B. Bachelor, et al., "A testbed for developing intelligent synthetic characters," in *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pp. 52–56, Palo Alto, Calif, USA, March 2002.

[7] R. Adobbati, A. N. Marshall, A. Scholer, et al., "Gamebots: a 3D virtual world test-bed for multi-agent research," in *Proceedings of the 2nd International Workshop on Infrastructure for Agents, MAS and MAS Scalability*, Montreal, Quebec, Canada, May 2001.

[8] S. Jacobs, A. Ferrein, and G. Lakemeyer, "Unreal Golog bots," in *Proceedings of Workshop on Reasoning, Representation, and Learning in Computer Games (IJCAI '05)*, Edinburgh, Scotland, UK, July-August 2005.

[9] J. E. Laird, "It knows what you're going to do: adding anticipation to a Quakebot," in *Proceedings of the 5th International Conference on Autonomous Agents*, pp. 385–392, Montreal, Quebec, Canada, May-June 2001.

[10] A. Khoo, G. Dunham, N. Trienens, and S. Sood, "Efficient, realistic NPC control systems using behavior-based techniques," in *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Palo Alto, Calif, USA, March 2002.

[11] T. S. Hussain and G. Vidaver, "Flexible and purposeful NPC behaviors using real-time genetic control," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 785–792, Vancouver, BC, Canada, July 2006.

[12] A. Nareyek, N. Combs, B. Karlsson, S. Mesdaghi, and I. Wilson, "The report of the IGDA's artificial intelligence interface standards committee," International Game Developers Association http://www.igda.org/ai/report-2005/report-2005.html, 2005.

[13] C. Berndt, I. Watson, and H. Guesgen, "OASIS: an open AI standard interface specification to support reasoning, representation and learning in computer games," in *Proceedings of Workshop on Reasoning, Representation, and Learning in Computer Games (IJCAI '05)*, pp. 19–24, Edinburgh, Scotland, UK, July-August 2005.

[14] A. BinSubaih and S. Maddock, "G-factor portability in game development using game engines," in *Proceedings of the 3rd International Conference on Games Research and Development*, pp. 163–170, Manchester, UK, September 2007.

[15] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, Reading, Mass, USA, 2001.

[16] T. Sweeney, "The next mainstream programming language: a game developer's perspective," in *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, p. 269, Charleston, SC, USA, January 2006.

[17] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice-Hall, Englewood-Cliffs, NJ, USA, 2005.

[18] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, vol. 1, John Wiley & Sons, New York, NY, USA, 1996.

[19] R. Bahsoon and W. Emmerich, "Architectural stability and middleware: an architecture centric evolution perspective," in *Proceedings of the 2nd International ECOOP Workshop on Architecture-Centric Evolution (ACE '06)*, Nantes, France, July 2006.

[20] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Proceedings of the 4th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '98)*, pp. 68–78, Monterey, Calif, USA, August 1998.

[21] P. Clements, "Active reviews for intermediate designs," Tech. Rep. CMU/SEI-2000-TN-009, Software Engineering Institute, Pittsburgh, Pa, USA, 2000.

[22] R. Bahsoon and W. Emmerich, "Evaluating software architectures: development, stability and evolution," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, p. 47, Tunis, Tunisia, July 2003.

[23] A. BinSubaih and S. Maddock, "Using ATAM to evaluate a game-based architecture," in *Proceedings of the 2nd International ECOOP Workshop on Architecture-Centric Evolution (ACE '06)*, Nantes, France, July 2006.

[24] A. BinSubaih, S. Maddock, and D. Romano, "Game logic portability," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '05)*, pp. 458–461, Valencia, Spain, June 2005.

[25] A. BinSubaih, S. Maddock, and D. Romano, "A serious game for traffic accident investigators," *International Journal of Interactive Technology and Smart Education*, vol. 3, no. 4, pp. 329–346, 2006.

[26] E. Dounis, "The great debate: gameplay vs. graphics," http://www.gamersmark.com/articles/205/, 2006.