Hindawi
Scientific Programming
Volume 2022, Article ID 1557302, 19 pages
https://doi.org/10.1155/2022/1557302

Research Article

Automatic Generation of Real-Time Animation Game Learning Levels Based on Artificial Intelligence Assistant

Rong Zhang

School of Artificial Intelligence, Dongguan Polytechnic, Dongguan, China

Correspondence should be addressed to Rong Zhang; 443798430@qq.com

Received 27 July 2022; Revised 7 September 2022; Accepted 19 September 2022; Published 13 October 2022

Academic Editor: Y. P. Tsang

With the development of computer animation games, now many animation games use the automatic generation of animation game levels method to generate animation game content. Using the automatic generation levels of the animation game method can quickly generate many different levels, saving labor costs in manual production. And with the combination of other fields, more accurate results can be obtained. In this paper, the goal is to automatically generate the level of the animation game. This research proposed animation game objects and victory conditions to derive the concept of the clearance path and uses artificial intelligence genetic algorithms to obtain the clearance path that best meets the needs of the developer to recombine new levels. In addition, since the generated levels are regenerated from the clearance path, there are still too many blank areas. In order to make full use of these blank areas, this research provides a level complete system. The system has formulated its placement rules for existing animation game objects. These rules fill in these blank areas and produce a more complete level. The contributions of this search proved that the genetic algorithm can converge quickly due to the correction effect in the mutation. In the complete system, it has successfully achieved a complete and playable level. In the system running time, this research adjusted the method of selecting elites and the use of permutations and combinations to mate, so that the system's operation can be stabilized on an average value.

1. Introduction

With the development of advanced animation games, more and more animation games also use ALG methods to generate animation game levels. Through ALG, different scene distributions can be easily explored. After the corresponding screening, a large number of qualified levels can be easily obtained, which greatly reduces the labor cost of manual production of levels and saves the storage capacity required for animation games. In Algwiki, this research has compiled a list of animation games using ALG technology, which contains many different types of animation games. There are single-player animation games and even multiplayer animation games. It can be seen that the development of ALG has gradually matured. And it also contains some animation game masterpieces that also use this technology. Therefore, it is also hoped that ALG technology will be introduced into the animation game, through the automatic generation of levels to generate many different levels, and to further complete the generated levels, so that the level can be improved and reduce the excessive probability of simple or unplayable levels. The article mainly consists of three parts: (1) create an animation game platform that can be loaded into a custom level for players to play, (2) research and develop an automatically generated level suitable for animation games and reduce the labor cost required for manual production of levels, and (3) provide a level complete system to correct oversimple or unplayable levels, and through corrections and adjustments, more different combinations of levels can be generated [1–3].

The goal of this research is to automatically generate the level of the animation game. First, we set the name and shape of the animation game object in the configuration file. The shape is represented by a 2D matrix, and then, we set the probability table to define the animation game object in the level. The position of the appear in the setting file is set in the

configuration file, which will be divided into three parts: upper, middle, and lower. After the setting is completed, the system will randomly generate a set of initial levels. This research uses the concept of clearance path. Applying this concept to the genetic algorithm, the system will obtain the clearance path data of these initial levels and pass the level. Crossover and mutation of the path generate multiple sets of new clearance path combinations. These new clearance paths can be reconstituted into new levels [4–6]. Finally, these new levels will be evaluated to obtain the corresponding scores, and the system will take them out according to the relevant settings in the configuration file. The corresponding number of levels will be included in the mating pool to participate in the next generation. Since the levels generated by the genetic algorithm have undergone many mating and mutations, the playability and completeness of the levels are not reliable. Therefore, this research provides a complete level system. After this system, this research aims at the level. The Rule-Base has been revised and adjusted, and multiple sets of different possible correction levels have been generated to obtain a more complete level [7–9].

The architecture of the whole system mainly consists of four parts. (1) Web server provides an API for better communication on the client side and the animation game database. (2) Animation game database mainly stores player information, models used by the level, and level information. (3) Client-side provides an animation game of unity of player version. (4) Generator provides automatic level generation and level integration system. As a contribution to the platform of the animation game, this research has implemented an animation game platform that can load custom levels for players to play and automatic levels generation based on AI genetic algorithms. Secondly, this research also proposed a level complete system to modify and adjust the generated levels. According to the experimental results, the completed level has a significant difference in the completeness of the scene compared with the incomplete level. Completeness still guarantees playability. Through the complete system, more different levels can be generated after the completion, so that developers can have more choices, also increase the system's exploration of levels, and increase the probability of the appearance of levels that meet the designer's needs [10–13]. The novel contributions of this search proved that the genetic algorithm can converge quickly due to the correction effect in the mutation. In the complete system, it has successfully achieved a complete and playable level. In the system running time, this research adjusted the method of selecting elites and the use of permutations and combinations to mate, so that the system's operation can be stabilized on an average value [14, 15].

2. Related Work

With the development of computer animation games, and in order to save labor costs, more and more animation games adopt ALG methods to generate animation game content, such as some classic animation games, elf, Angry Birds, and Diablo [16–18]. Therefore, this research also hopes to introduce ALG technology into the animation game.

2.1. Hyperautomation. Hyperautomation is a true digital transformation with the help of advanced techniques such as Robotic Process Automation (RPA), Machine Learning (ML), and Artificial Intelligence (AI). It automates complicated business processes, even where topic specialists were formerly needed. This is an expansion of the processes of traditional business process automation. Hyperautomation allows automation to do virtual tasks performed by business people by merging AI technologies with RPA. This takes us to the next level for detecting and generating automation processes dynamically. It allows companies to combine business intelligence systems, undertake complex needs, and increase human expertise and automation experience. This paper briefly discusses hyperautomation and its need in the current scenario. Then, it elaborates the significant roles of sensors to enhance hyperautomation. Various versatile technologies, such as dedicated workflow processes and specific domains of solicitations associated with hyperautomation, are also discussed diagrammatically. Then, this study further identifies and discusses the capabilities of hyperautomation for industries. Hyperautomation is being utilized to increase the efficiency and human enhancement of automated operations substantially. It comprises several automated tools, including analytics, discovery, design, measurement, monitoring, and complex automation components. Thus, it ideally utilizes to integrate state-of-the-art tools and develop new methods of working [19–21].

Robotic Process Automation (RPA) deals with the automation of rule-based process tasks to increase process efficiency and reduce process costs. Due to the utmost importance of business process automation in industry, RPA attracts increasing attention in the scientific field as well. This paper presents the state of the art in the RPA field by means of a Systematic Mapping Study (SMS). In this SMS, 63 publications are identified and analyzed. From the SMS findings, a framework for systematically analyzing and comparing RPA works is derived. The discovered thematic clusters suggest further investigations to develop a more elaborated structural research approach for RPA [22–24].

The vision of factories of the future is motivating many industrial companies to modernize their existing portfolio of systems and services to maintain market share and improve business agility. For long-lived industrial systems, it is challenging to adapt legacy assets to a service-oriented stream in cloud computing and Internet-of-things contexts. For this reason, many research studies have proposed techniques and methodologies to migrate legacy industrial functions and systems at different hierarchy levels of automation control. This paper presents an overview of these techniques and methodologies, as well as industry practices to achieve the vision of factories of the future. A better understanding of the challenges encountered in legacy migration processes will help researchers and practitioners in their further efforts [25].

2.2. Animation Game. The background of the animation game came to the world and learned that a few behaved apples lost. He had to send the lost apple home, but he would

encounter them on the way. When you go to some terrible institutions is different from the traditional puzzle-solving animation game, which only controls a single object to perform the animation game, such as traditional classic puzzle-solving animation games such as warehouse fan, push music, Mario and Tetris, and a simultaneously an animation game that controls two objects to complete the animation game level and combines physics to make collision behaviors with other objects. In the animation game, players use both hands to control two s at the same time and transport the apple back to the bag [26–29]. Completing the animation game level requires not only good control ability but also excellent hand-eye coordination [30–32].

Players operate two s in different directions to move in two directions (left and right). In the level, by controlling the transported to the apple bag, then you can enter the next level. The player can use the keyboard keys *Z*, *X* and *N*, *M* to control two s at the same time. There will be some obstacles in the process of apple to prevent the player from delivering the apple. These obstacles are as follows:

Gap: it may cause the apple to fall on a more difficult route, and the player must accelerate across the gap or may need to swap the positions of the apple to reach the gap smoothly.

Trap ball: it may swing around the player's path. When the apple hits the trap ball, it will be knocked away. The player must calculate the swing time of the trap ball or use collide the trap ball multiple times to get the trap ball [33].

2.3. Automatic Level Generation (ALG). This article mainly uses genetic algorithms to automatically generate animation game levels. The following are related documents for automatically generating animation game levels: the method proposed by Ferreira (Ferreira and Toledo, 2014) uses genetic algorithms to generate Angry Birds levels, treats all elements in the Angry Birds levels as an object, introduces the object into the concept of stacking, and proposes a fitness function to evaluate the balance and quantity of the level elements. This method can generate levels in real time, and the generated levels can be balanced without collapsing. The method proposed by Snodgrass cuts the map of Super Mario Bros into H∗W Tiles, given some existing Mario Maps, learns the occurrence probability of elements in the map through Markov Chains, and finally uses these chances to generate new Mario Maps. The method proposed by Dahlskog divides the map of Super Mario Bros into three different levels of patterns, namely, Micropatterns, Mesopatterns, and Macropatterns. Mesopatterns are composed of Micropatterns, and Macropatterns are composed of multiple Mesopatterns. The Mesopatterns have different classifications, including Enemies, Gaps, Valleys, Multiple Paths, and Stairs. Finally, through Evolution Algorithm, Micropatterns are combined to generate new Mario Maps, and meaningful combination levels are left through the fitness function. The method proposed by Horn and Dahlskog uses a variety of related Mario AIs that have been published and sets different indicators to obtain the different characteristics of these AIs. Finally, these characteristics can be used to generate levels. It is also possible to combine multiple different AI features to obtain a better level or meet the designer's expectations. This research refers to the method proposed by Ferreira. The level is cut into a 32∗24 Grid, and the concept of the customs clearance path is derived based on the relationship between the elements and the customs clearance conditions. The concept is brought into the genetic algorithm and then applied to the automatic generation of animation game levels. Different levels can be obtained by controlling the system parameters and the probability table of the relevant element data and the element placement position for a given level.

Also, levels are generated using 16 patterns in the system to combine multiple levels with different configurations. The difference from this is that this system only needs to set the system parameters and the animation game object and the animation game object placement probability table to generate the level. Its ALG has been used in many different types of animation games. In the article "The Death of the Level Designer," the ALG used in animation games is subdivided into the following categories: runtime random level generation, design of level content, dynamic world generation, instancing of in-game entities, user mediated content, dynamic systems, procedural puzzles, and plot generation from the above classification; it can be seen that ALG has been flexibly used in animation games, such as animation game content, maps, and animation game plots. In addition, Togelius also mentioned that the production methods of ALG can be divided into Online or Offline, Necessary or Optional Content, Random Seeds or Parameter Vectors, Stochastic or Deterministic Generation, and Constructive or Generate-and-Test. ALG is also integrated with other fields and therefore has more room for development and applications, such as Learning-Based and Search-Based. The following will explain the related documents of automatic level generation and genetic algorithm, respectively [34, 35].

2.4. Genetic Algorithm (GA). A genetic algorithm (Genetic algorithm, 2017) is developed based on the evolutionary phenomenon of the biological world, which includes heredity, mutation, natural selection, mating, etc. This research uses computer simulations to implement this algorithm. This research treats parameters as genes, allows these genes to evolve, and then explores and obtains parameters that are more suitable for solving the problem. Genetic algorithms are good at solving global optimization problems. For example, to solve the problem of scheduling timetables. Many types of software for scheduling timetables use genetic algorithms. Nowadays, genetic algorithms are also used in many different fields, such as circuit design and neural networks [36, 37] for manufacturing systems/processes [38, 39]. This paper derives the concept of the customs clearance path based on the animation game clearance conditions, uses genetic algorithms to generate better clearance paths, evaluates these clearance paths through the evaluate function, and retains the levels that meet the expectations. As the generations evolve, we will get the most consistent level of evaluation [40–43].

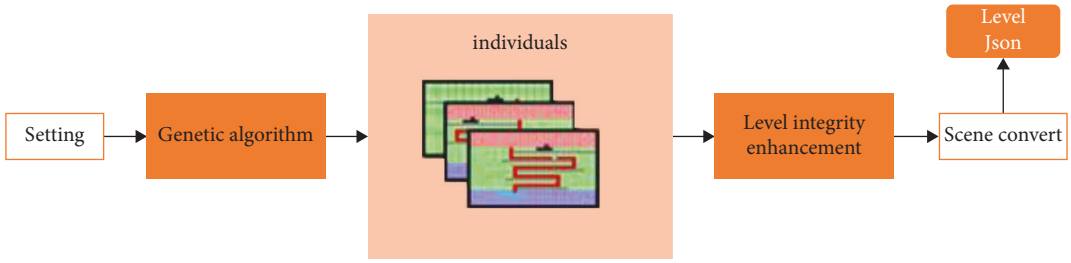


Figure 1: The framework of our proposed method.

Table 1: System basic parameter setting.

Main Main setting	width	Grid preset width
	Height	Grid preset height
	Top	Defines the top area
	Bottom	Defines the bottom area
	init_gen_num	Initialize the number of genes
	Population_Elitism	Parent selected number
	Elitism	Mating pool elite selection (%)
Ground initialize ground-related settings	min	Ground minimum length
	max	Ground maximum length
	Count	Ground out of the count initialization level quantity (min, max)
Trap Trap ball-related settings	max_count	Maximum number of trap spheres
Export output conversion-related settings	save_path	Output file placement location
	background_res	Unity_prefab related path settings
	Prefab_psth	
	_Res	
	floors_res	
	apple_res	
	bag_res	
	trap_res	

3. Research Methods

This article mainly uses genetic algorithms to generate animation game levels; at the same time, a level complete system is made to reduce the blank area in the level, so that the level can be closer to a complete level, as shown in Figure 1. The method can be divided into two parts: (1) genetic algorithm and (2) level integrity enhancement.

3.1. Genetic Algorithm. This article is inspired by Ferreira, referring to its system architecture combined with genetic algorithms to generate animation game levels, and uses the path of the level as the main axis of the level to ensure that the level can be cleared. This section disassembles the genetic algorithm into multiple small sections to explain it one by one [44–47].

3.1.1. Scenario Structure. In this article, referring to the concept of level division in Ferreira (Ferreira and Toledo, 2014), the level scene is vertically divided into three parts, the top, the middle, and the bottom. This research regards the level as a grid, and this grid has a total of 32∗24 Cells, and its animation game screen resolution is preset to 1024∗768; that is, each cell occupies 32∗32 animation game screens. *Pixel.* The entire level is divided into three layers. From top to ground, there are three layers: top (red area), middle (green area), and ground (blue area); top and ground are set by parameters in the configuration file, and the remaining areas belong to the middle.

3.1.2. System Parameter Setting. Before the system starts to operate, some necessary parameters must be set so that the system can execute smoothly and obtain results that meet the expectations of users. There are 3 configuration files in the system. Table 1 is the basic parameter settings of the system. The parameters set here are the system defaults.

Since the system uses the grid to represent levels, it needs to put the objects in the animation game level as grid representation as Figure 2 and separately describe the cell occupancy pattern of its animation game object in the grid.

The representation method is shown in Table 2. There are 2 types of representation methods for the area description. (1) The area description is N or a number, which means the X-axis length of the object is N. (2) Take the object as an example; the area description is 4∗2, which means the X-axis length is 4, and the Y-axis length is 2 for the object. If the area is described in the above-mentioned type 2 notation, the pattern must be defined separately. In the pattern, 0 represents a blank area, 1 represents the physical area of the

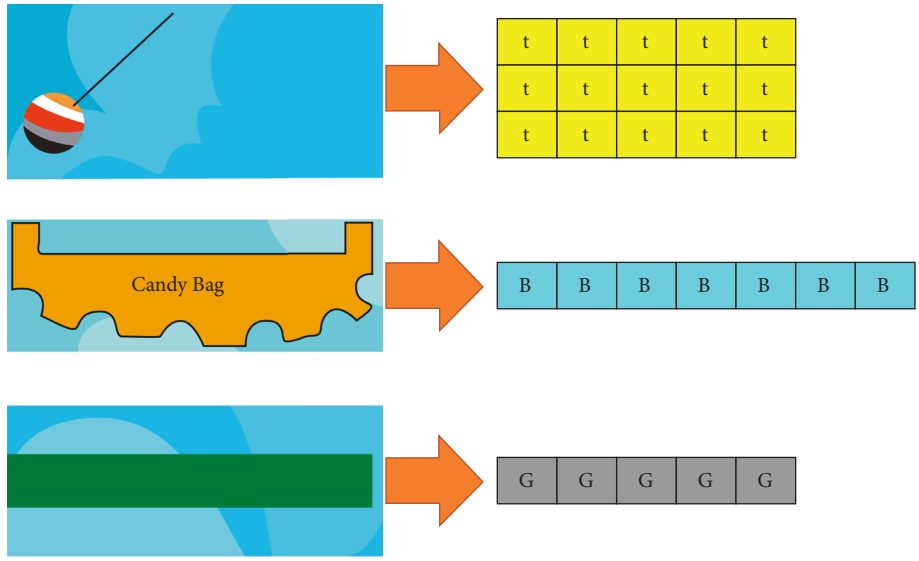


Figure 2: Animation game object pattern.

Table 2: Animation game object setting table.

Ground	G	N	1
Candy	C	1	1
Road	$p1$	$4 * 2$	[0, 0, 1, 0], [1, 1, 2, 1]
Trap	t	$5 * 3$	[1, 1, 1, 1, 1], [1, 1, 1, 2, 1], [1, 1, 1, 1, 1]
Bag	B	7	1
Empty	E	1	1

object, and 2 represents the reference point for placing the object when the system determines that an element is placed at a certain coordinate, its coordinate is the reference point of the element, and the Y-axis separation is indicated by a semicolon.

3.1.3. Probability Table Setting. In the initial population phase of the genetic algorithm, the system will refer to the probability set here to place the corresponding animation game object. Here, the probability normalize is a value of 0~1 (0%~100%), and the sum of the three floors' probability of a single object must be equal to 1.

3.1.4. Initialization. Since when the system is first executed, there are no individuals in the mating pool that can mate with each other, so it is necessary to initialize some available individuals to allow the system to execute smoothly. First, obtain the configuration information of each element through the probability table and randomly place it at a corresponding position in a random number manner to generate N individuals, where N refers to the init_gen_num parameter in the system setting, and the rules of ground will also be based on the system setting. The parameters are set. Since the animation game uses gravity physics, the individuals generated will first go through the gravity function. This function will give all elements except the floor ground the gravity effect so that the individual elements are placed in a position that conforms to the gravity effect. After the gravity function, a clearance path search will be performed to obtain the clearance path and apple information of the individuals. If the clearance path or apple information cannot be successfully obtained, the level will be determined as unavailable. Regenerate a new individual until the value of the init_gen_num parameter in the system setting is satisfied.

3.1.5. Clearance Path Search. The clearance path search function is mainly used to obtain the clearance path and apple information. First, the system will first obtain the bag element, take the cell occupied by the element as input, and perform reverse path tracking one by one. The system coordinate system here is shown in Figure 3. Taking a 32∗24 Grid as an example, the upper left cell coordinate is (0, 0), and the lower right cell coordinate is (31, 23).

The tracking method of the clearance path mainly uses two methods.

Method 1 is responsible for the Y-axis search. Each time it is executed, it will perform an upward search action (Method 1, line 5) until it encounters the ground (G) element or exceeds the coordinates outside the scene end.

Method 1: Scan Y

(1) *cell* ← [X, Y];
(2) while *cell.Y* != "*G*" *and cell.Y in Scene* do
(3) if (*cell*.LowerRight = "*G*") ScanX(*cell*, *Right*);
(4) else if (*cell*.LowerLeft = "*G*") ScanX(*cell*, *Left*);
(5) *cell.Y* ← *cell.Y*—*1*;
(6) end while

Method 2 is responsible for the X-axis search. It will determine the calculation of X according to the direction parameter (Method 2, line 2) every time it is executed and search to the left or right until there is no ground (G) element support below the position or out of the scene The coordinates are over.

Method 2: Scan X

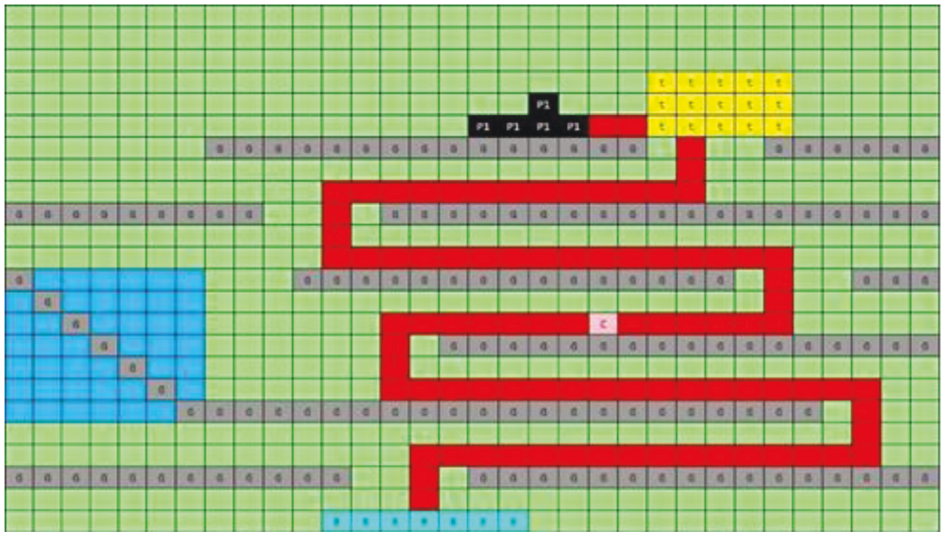


Figure 3: The result of placing the trap ball.

(1) *cell* ← [*X*, *Y*];
(2) *direction* ← (Left or Right)
(3) while *cell.down* = *"G" and cell.X in Scene* do
(4) ScanY(*cell*);
(5) if (*direction* = *Left*) *cell.X* ← *cell.X—1*;
(6) else if (*direction* = *Right*) *cell.X* ← *cell.X* + *1*;
(7) end while

After the above two methods, the blue path will be judged by the system as the customs clearance path.

3.1.6. Evaluation. The fitness function is used to find the optimal path from the source node to the destination node to reduce the energy consumption in multipath routing. The generated levels will be scored through the evaluate function (1). Here, the information of the pass path (clearance path) and apple obtained after the clearance path search is used.

Equation (1) is the fitness function (Ttito and Huacho, 2021):

$$f_{\text{ind}} = \frac{f_1(f_2 + f_3)}{2}. \tag{1}$$

Equation (2) ensures that apple exists:

$$f_1 = \begin{cases} 1, & [\text{apple}] \neq NULL \\ 0, & \text{Otherwise} \end{cases}. \tag{2}$$

Use (2) to ensure the existence of the apple. If the apple information is empty, the score will be 0.

$$f_2 = \frac{\text{Count}(\text{PassPath}, \text{CheckPoint})}{\max(\text{PassPath}, \text{CheckPoint})}\left(1 - \frac{|\text{Left.count} - \text{Right.Count}|}{\text{Count}(\text{PassPath}, \text{CheckPoint})}\right). \tag{3}$$

Equation (3) is the clearance path complexity.

In (3), this research uses the maximum Check Point of the level (equation (4)) and the number of Check Points currently owned by the clearance path as weights. The maximum Check Point uses the scene height to accommodate several s as the value, and the height of the element is increased by 1. It is due to the placement of the floor elements, and the final maximum value minus one is to deduct the reserved layer of the bottom bag.

Equation (4) is the Check Point Maximum:

$$\max(\text{PassPath}, \text{CheckPoint}) = \frac{\text{Scene.Height}}{\text{Monkey.Height} + 1} - 1. \tag{4}$$

Secondly, use the Check Point with the left direction and the Check Point with the right direction to calculate, so as to obtain the average result of the left and right times of the Check Point; that is, the left and right changes are more, which can increase the complexity of the customs clearance path.

$$f_3 = \frac{\text{PassPath}_{\text{countmovement}}}{\text{PassPath}_{\text{totallength}}}. \tag{5}$$

Equation (5) obtains the percentage of the total length of the clearance path and the number of steps that can be moved as the scoring benchmark. The number of steps that can be moved is defined by the support of the floor element under this coordinate. The higher the score, the longer the

length and the movable step. The number is large, so as to obtain a longer customs clearance path.

3.1.7. Selection of Mating Group. Here, the mating chromosomes will be selected for genetic evolution. After evaluation, all individuals will have a score, which will be ranked from high to low according to the score, starting from the highest score, and M individual individuals will be taken out, where M is the Population Elitism parameter in the system setting and will be selected. The individual will be put into the mating pool and used by crossover and mutation later.

3.1.8. Mating. The individual system selected by a select population takes its clearance path as the chromosome. This research uses the permutation formula P_n, where n is the population Elitism parameter in the system setting. Arrange every 2 clearance paths in the list as a group and each group as a uniform crossover. Take an example, the two sets of chromosomes have different lengths, and both have the first three sets of genes, then the first three sets of genes have a 50% chance to exchange each other, and the first set of chromosomes lacks the fourth and fifth sets of genes. There is a 50% probability that the second set of genes will be copied. The example is the result of copying the last set of genes. Since there may be genes that can be merged after the crossover, it uses the merge method to integrate the genes. Crossover here uses a small number of high-scoring maternal populations to exchange their gene combinations to generate chromosomes with a larger number of genes or different gene combinations. Secondly, the number of genes can be increased to increase the probability of a better combination.

3.1.9. Merge Method. After the crossover, the gene system will be in the chromosome, and the two adjacent genes will be checked one by one in a group. The judgment conditions are as follows:

(1) The same direction (←, ← OR ⟶, ⟶ OR ↑↑) will synthesize a group of genes

(2) The opposite direction (←,⟶ OR ⟶,←) values cancel each other and finally combine into a group of genes

(3) If the directions are not the same and one group of genes is "↑", the next group will be judged directly

3.1.10. Mutation. In the crossover part, it is mainly by exchanging genes to generate more possible combinations and chromosomes with longer gene lengths. Make rational corrections and probability mutations for each group of genes to obtain more different possibilities, so as to explore possible better results. The mutation of this system is mainly divided into four parts. Among them, direction mutation and Check Point mutation are probability mutations, and fixed length and fixed path are rationalization correction directions,

3.1.11. Direction and Check Point Mutation. The first part of gene mutation is the chance of changing the direction of the gene. The mutation probability of each group of genes is 50%. The system will replace the originally left gene with the right gene with a 50% probability. If it is an original right direction gene, it will be replaced with the left direction, and if it is an upward direction gene, the system will obtain the remaining blank area above the scene to randomly change the upward direction value but will not change the upward direction.

The second part of the mutation is for genes in the upward direction. The system will base on the height of the (player) element. If a group of genes with an upward direction is found to be higher than the height of the element, the system will respond with a 50% probability. Splitting the upward gene will split the upward gene into two groups of upward genes; inserting a set of random left or right genes between the two groups of upward genes, this action will increase the coloring of the number of genes in the body used to increase the complexity of the customs clearance path. An example of gene splitting: the genes marked in red font on the left are split into 3 genes on the right.

3.1.12. Correction Function. Here, the system will perform correction actions for each group of genes. The main purpose is to make more combinations that were previously unavailable into useable combinations and to explore better combinations through corrections. The system will make correction judgments for each group of genes one by one. The correction is divided into two parts; the first part is the length correction. The system will adjust the genes that do not meet the basis value based on the width and height of the (player) element. The gene of the basis is given a value that satisfies the basis again. The value basis for the left and right directions is 6, and the value for the upper direction is 3. If the value in the left and right direction is less than 6 and the value in the up direction is less than 3, a new value will be assigned.

The second part is the direction modification. After crossover and mutation, this research found that many levels have undergone the step of modifying the gene length. Although the goal of increasing the length has been achieved, the path of clearance may go beyond. Therefore, the system calculates according to each group of genes and at the same time checks whether the calculated result is out of the scene; if it is out of the scene, replace it with the gene in the opposite direction as a correction.

3.1.13. Selection of Elites. After crossover and mutation, the system ranks the scores of all individuals from high to low. Refer to the Elitism score threshold parameter in the system setting to calculate P_{score}. The formula is as follows.

Equation (6) is the Elitism score threshold calculation formula:

$$P_{\text{score}} = \max(\text{mathingPoolScore}) - \frac{\max(\text{maxtingPoolScore}) * \text{Elitism\_score\_threshold}}{100}. \quad (6)$$

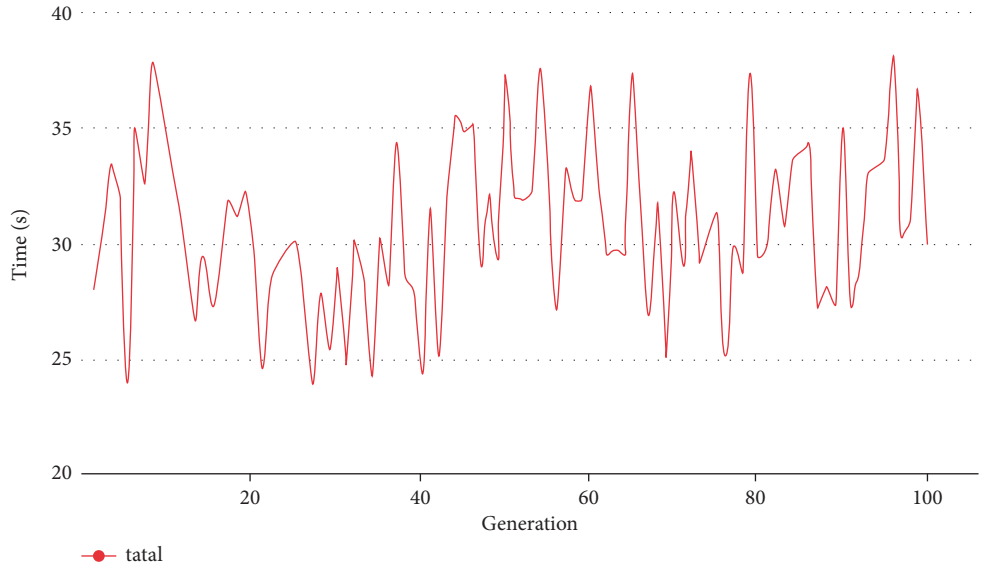


Figure 4: Line chart of the total time spent in the execution of the 100-generation system.

If the current level score is higher than P_{score}, the level will be transferred to the mating pool to participate in the next generation; otherwise, the level will be discarded after all actions of the generation are over.

3.2. Level Integrity Enhancement. After the genetic algorithm, it can obtain multiple sets of levels, and these levels all have a path to pass the level in Figure 4 (red path) that can guarantee its playability, but this research found that there are still many unused levels in these levels. The entire level does not look complete enough. Therefore, it has developed a system to strengthen the integrity of the level and set a number of rules to fill in the blank areas in the level and increase the diversity of the level.

3.2.1. Completion of Customs Clearance Path. The clearance path is the main reference core. The purpose of this research is to ensure that the clearance path is unobstructed, which means that the level is playable. The level generated by the genetic algorithm level generation system is based on the relative position of the clearance path endpoint and determines the generation of the top floor, so the system first connects to the end of the clearance path. It needs to define a complete clearance path. The clearance path is used to ensure level playability, and the clearance path is blocked or covered arbitrarily. Such actions mean the need to bear the risk of reduced playability.

3.2.2. Floor Filling. After completing the clearance path, the system will then fill in the blank area in the level. First, you need to analyze the level and divide the floors in Figure 5. If ground appears on the Y-axis of the scene, the system will regard it as a floor, and the floor where the element is located is the top floor, and the bottom floor is the bottom floor. The top and ground floors are called for the middle layer.

After the floor is cut, the system will be divided into two parts for processing. The top part of the system will detect the direction of the path through the customs and open a hole on the other side of its direction. This action can increase different path lines so that this level has more than a single path to choose from at the beginning of the game. For the middle layer and the bottom layer, each layer will fill its ground but retain its clearance path.

3.2.3. Ensuring Customs Clearance Path. After the system fills up each floor, although the clearance path is still maintained, the gap must be greater than the width of the clearance path to be truly effective. Therefore, the system will check the width of the gap in the clearance path here to ensure that the clearance path can really let the element pass. First, the system will first obtain all the coordinates on the path of the clearance, and each coordinate will return a set of 9 grids pattern; this research does individual processing for 3 different patterns.

The center point of all patterns is the detection point. If the pattern belongs to Figure 6(a), the system will convert the 2 cells on the left and the 2 cells on the right of the detection point into blank blocks to ensure that the element can be smoothly followed. The clearance path drops down. In Figure 6(b), the system converts 1 cell on the left and 2 cells on the right of the detection point into blank blocks.

3.2.4. Placing a Sloped Floor. In order to increase the diversity of the animation game, this research added a sloped floor, and because it has a gravity physics system, adding a sloped floor can also relatively improve the animation gameplay and difficulty. In order to avoid the problem of the joints of the diagonal floor and whether they can go up to the slope, the sloped floor is divided into two types: single-joint slope and double-joint slope. The single-joint slope in Figure 6(a) is classified in this research. It is defined as a 30-

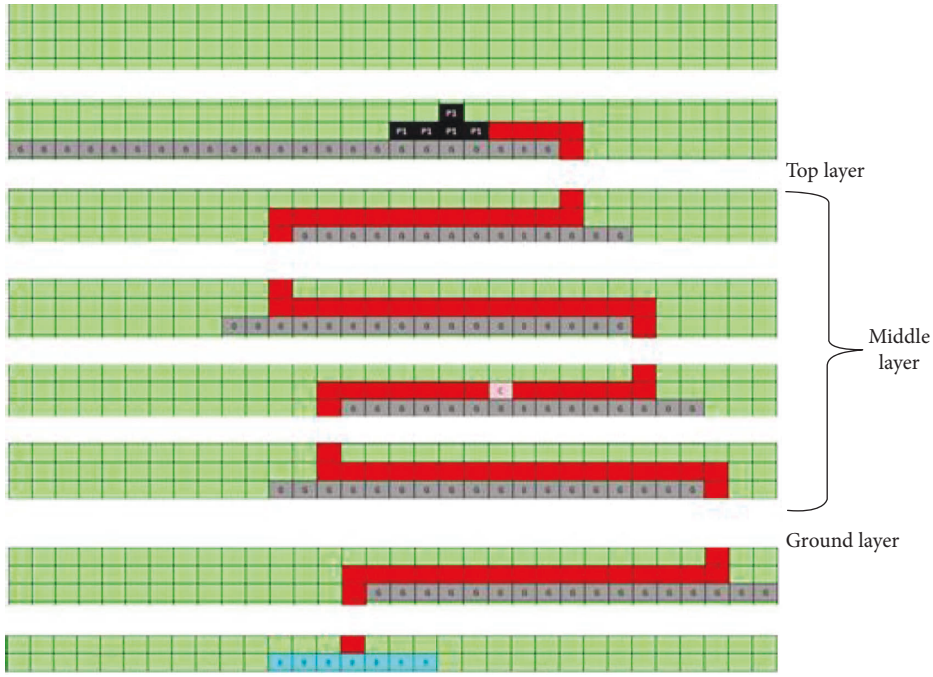


Figure 5: Floor cut plan.

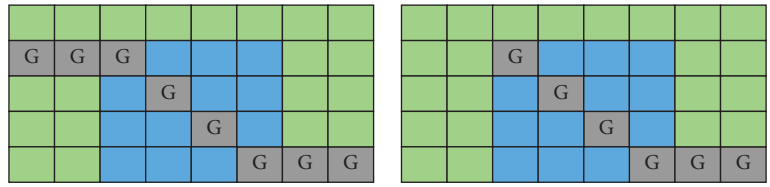


Figure 6: Sloping floor. (a) Double-joint slope. (b) Single-joint slope.

degree slope. It has been tested that can climb uphill, but the 45-degree slope cannot be climbed because it is too steep. The double-joint slope in Figure 6(b) system uses a 45-degree slope, which means it cannot climb.

Next, the system will detect the area that is not affected by the clearance path on the premise where it does not affect the clearance path, and treat this area as a variable area; that is, slope elements can be placed. The method is to detect the shortest distance between the left and right edges of the scene and the clearance path to infer the variable area. Therefore, this research will obtain the minimum X (CMin) and maximum X (CMax) two parameter values; the formula is as follows.

Equation (7) is the calculation formula for variable area:

$$\begin{aligned} \text{CMin} &= \min_{\text{PassPoint}\in\text{PathPass}} (\text{PassPoint}.X) \\ \text{CMax} &= \text{Scene.width} - \max_{\text{PassPoint}\in\text{PathPass}} (\text{PassPoint}.X) \end{aligned}. \tag{7}$$

The value of CMin is the minimum X value of Pass Point on all clearance paths, and the value of CMax is taken as Pass Point. The maximum X value is subtracted from the scene width. Taking Figure 7 as an example, CMin is 11 and CMax is 2.

Because the sloped floor element will occupy a large area, and it is easy to break the rules of the element arrangement in the scene, the system will calculate the clearance path as the center and determine which side of the scene is the largest area and then place the sloped floor in the largest area. The method is to take the middle layer and the bottom layer and calculate the moving cell of each layer from the left and right edges of the scene to the customs clearance path. The formula is as follows:

$$\begin{aligned} \text{Right} &= \sum_{n-1}^{\text{cnt}} R_n \\ \text{left} &= \sum_{n-1}^{\text{cnt}} L_n \end{aligned}. \tag{8}$$

Equation (8) is the formula for calculating the azimuth of the largest open area in the scene. In the formula, cnt is the total number of the middle layer and the bottom layer, L is the cell occupied by each layer from the left of the scene to the customs clearance path, and R is the cell occupied by each layer from the right to the customs clearance path. The

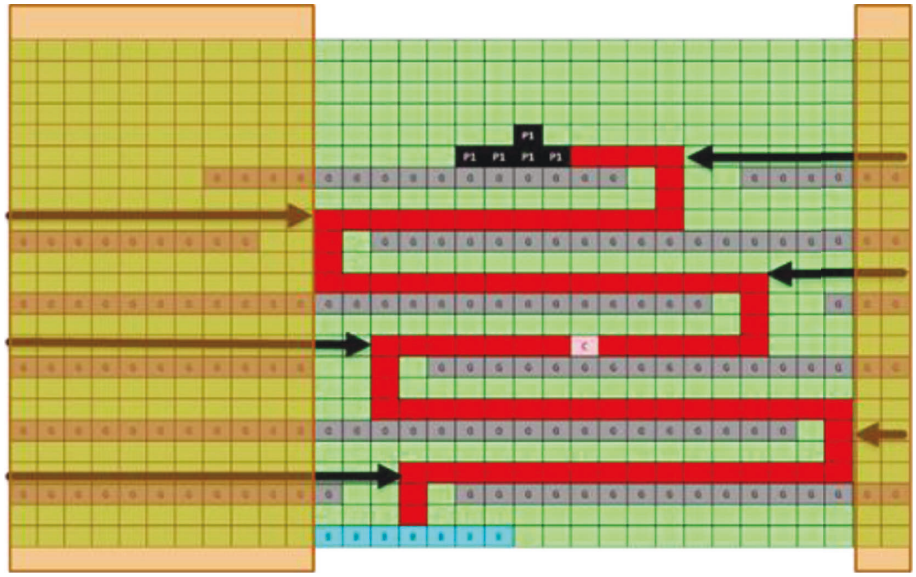


Figure 7: Schematic diagram of the variable area.

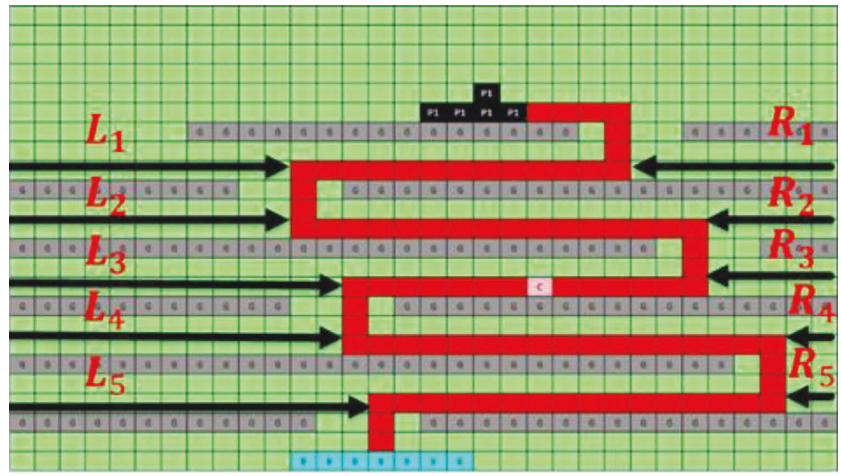


Figure 8: Schematic diagram of calculating the position of the largest open area in the scene.

schematic diagram is shown in Figure 8. After calculating the left and right values, the system will select the larger value as the placement direction.

After obtaining the position of the largest open area and the variable area of the scene, the system will randomly select the ground element of the two floors in the middle layer, namely, two Y, in the position of the largest open area of the scene. The X calculation formula is as (9) slope placement point X value calculation, connecting two points, namely, the slope floor entity; when placing the slope, the system will place the nonphysical area in a square area as a blue block, as shown in Figure 9.

Equation (9) is the calculation of X value of slope placement point:

$$X = \begin{cases} \text{Left} > \text{Right}, & X < \text{Min} \\ \text{Right} > \text{Left}, & X > \max \end{cases}. \quad (9)$$

3.2.5. Placing the Trap Ball. The trap ball element is added to the level here. The system will search for the hole on each floor and randomly choose a place close to the ground at the hole to place its trap ball. This research believes that proximity to the ground element can increase the chance of contact with the player. Regarding the scene, the maximum number of trap balls is set in the trap.max_count parameter in the system setting. After placing trap balls, the result is shown in the yellow area in Figure 3.

3.2.6. Opening the Second Path. This research has successfully added two new elements (slope floor and trap ball), but because the current level mainly retains the ground element according to the path of the level, the path of the level is easy to be seen by the player at a glance, so the system provides a method to open different routes from floor to floor to confuse the customs clearance route, which is to improve the players in each floor to have left and right routes to choose

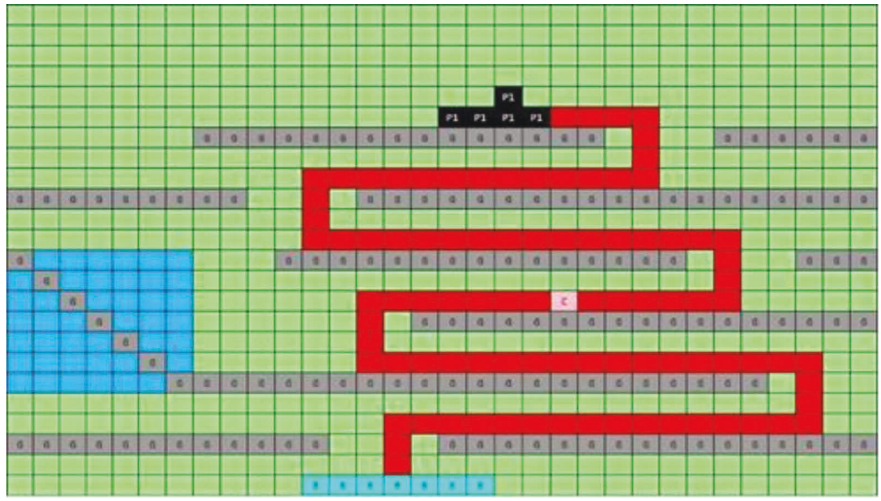


Figure 9: Result of placement of sloping floor.

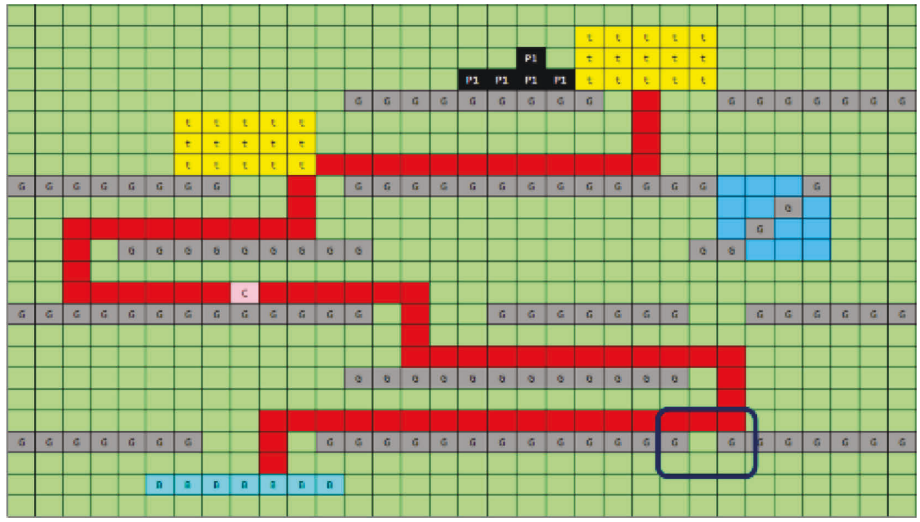


Figure 10: Example of obstructing apple.

from instead of just the clearance route options. The system will take out the *Y*-axis position of all floors and open the second route in the middle layer one by one. The system will start from the top layer, and there will be one set for every three floors. If there is ground on the first and third floors, delete it. The second layer is the ground. Based on experience, this research found that the level route can be complicated to an S-shaped route, which is also the main concept of this method.

The following randomly takes out an area in the scene as an example. If the floor structure of this area is not the same, it is divided into two blocks for illustration. Small blocks will be found. The ground element is in the second step because the ground element is bound by the clearance path (red square); that is, the second route is still based on the principle of not destroying the clearance path. Refer to the process. After opening the second route, the checkpoint result appears.

3.2.7. Apple Trap Settings. In order to make the animation game a little more difficult, the system provides a method to hinder the transfer of apple. The system first obtains the floor position of the apple element in the level, from the ground floor to the floor where the apple exists, searches for the nonfloor cells of the floor, and obtains the nine-square grid pattern of these nonfloor cells one by one. If the pattern meets the rule in Figure 3, calculate its landing point and create a small hole below the landing point in Figure 10. This hole only allows the apple element to pass, and the element can pass through the hole when accelerating to a certain speed. The existence of this hole causes the player to control the impact. Try to keep the apple element from falling into a hole to increase the difficulty of the animation game.

3.2.8. Level Floor Correction. Since the system has some damage to the distribution of level elements after the actions in the above few subsections, the system will modify the distribution of ground in this step. There are three main processing steps for the modification. The first step is to correct the floor at the edge of the scene in Figure 11. The system will ensure that the width of this element can be used by the (player) element from the ground elements on each

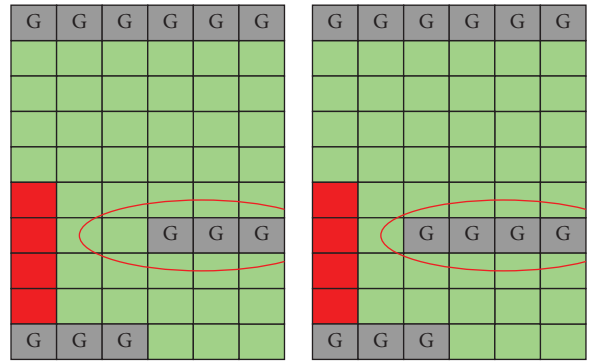


Figure 11: Correcting the edge floor. (a) The length of the edge floor before correction is only 3, which cannot accommodate the length of the element. (b) After correction, the length is 4, which can accommodate elements.

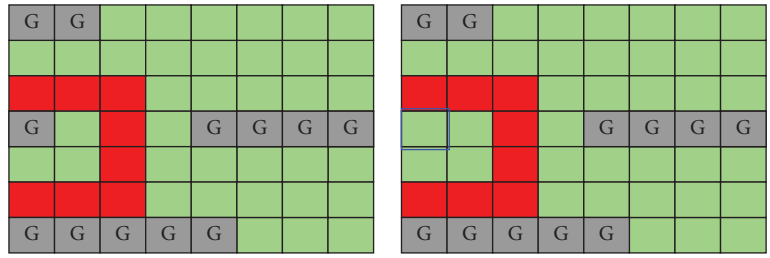


Figure 12: (a) Clearance path hole correction. (b) After the correction, the system decides to keep the edge and delete the floor element in the opposite direction to make the hole. The width of the mouth satisfies the element. Finally, the system will search for all floor elements on the level and check whether it is greater than the width of the element. Floors that are less than the width will be filled to the required width to reduce the existence of unusable floors. The edge floors will also be avoided here.

floor on the left and right sides of the scene. The purpose is to make the floor at the edge of the scene more meaningful. The floor, that is, the element, can stay.

Second, the system will modify the ground element of the landing point position of the customs clearance path to ensure that the landing point of the customs clearance path can allow the (player) to land normally as in Figure 12. At the same time, the system will also ensure that the customs clearance path is smooth, that is, each Pass Point. The sum of the left and right widths must be greater than or equal to the width of the element. This step ensures that the level is playable in Figure 12.

3.3. Output. After the level is complete, the level must be converted to a level file. First, coordinate conversion is required. In the system, its (0, 0) position is in the upper left corner of the cell, but it is at the center of the scene, and the length and width of the elements refer to the width and height in the system setting and the level resolution size (1024∗768) preset and take the aspect ratio value for calculation; then, the conversion can be successful. Regarding the seams of sloped floors, this article also provides a set of equations to calculate the correct placement coordinates. To use this equation, the sloped floors must be divided into two categories, the slope of the single joint is 30 degrees, and the slope of double joints is 45 degrees. For the single-joint slope part, first, convert the slope floor to the animation game coordinates, calculate the X and Y correction values in equation (10), and then add and subtract actions according to the joint points in Table 3 to smoothly change the slope. The floor is connected to the horizontal floor at the joint point.

Equation (10) is the coordinate adjustment of slope floor output conversion:

$$\begin{aligned} X_{\text{fix}} &= 0.683 + (0.433 * (\text{slash.length} - 1)) \\ X_{\text{fix}} &= -0.183 + (-0.25 * (\text{slash.length} - 1)) \end{aligned}. \tag{10}$$

The double-joint sloped floor rotates the horizontal floor by plus or minus 45 degrees, and this article solves the joint problem by calculating the correction value based on experience as shown in Table 4. Under this formula, the placement coordinate of the sloped floor is the center position of the two joints and its coordinates.

4. Experimental Results

4.1. System Architecture. The animation game system architecture in Figure 13 is mainly divided into four parts, including (1) server, (2) animation game database, (3) client, and (4) generator. Among them, the web server mainly

Table 3: Conversion table of single-joint slope correction coordinates.

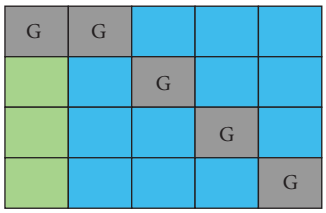
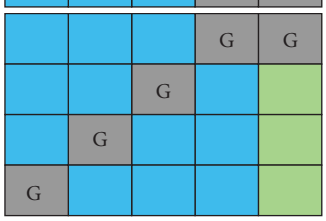
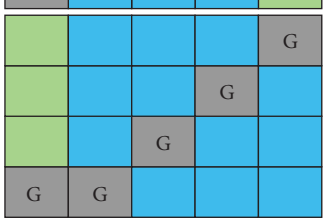
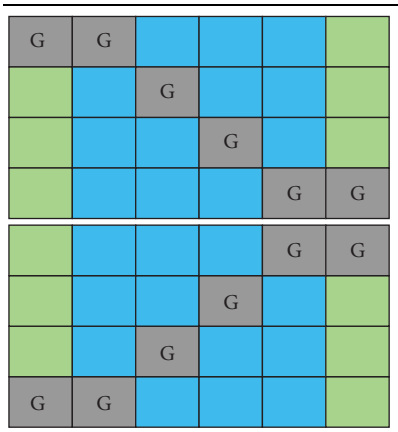
	$Xcur = X + X_{fix}$
	$Ycur = Y - Y_{fix}$
	$Xcur = X - X_{fix}$
	$Ycur = Y + Y_{fix}$
	$Xcur = X - X_{fix}$
	$Ycur = Y - Y_{fix}$
	$Xcur = X + X_{fix}$
	$Ycur = Y + Y_{fix}$

Table 4: Corrections.

	$Length_{cur} = 4.5$	$Xcur = X + 0.7375$
		$Ycur = Y - 0.0455$
		$Xcur = X + 0.2375$
		$Ycur = Y - 0.0455$
	$Length_{cur} = 8.5$	$Xcur = X + 0.65$
		$Ycur = Y - 0.145$

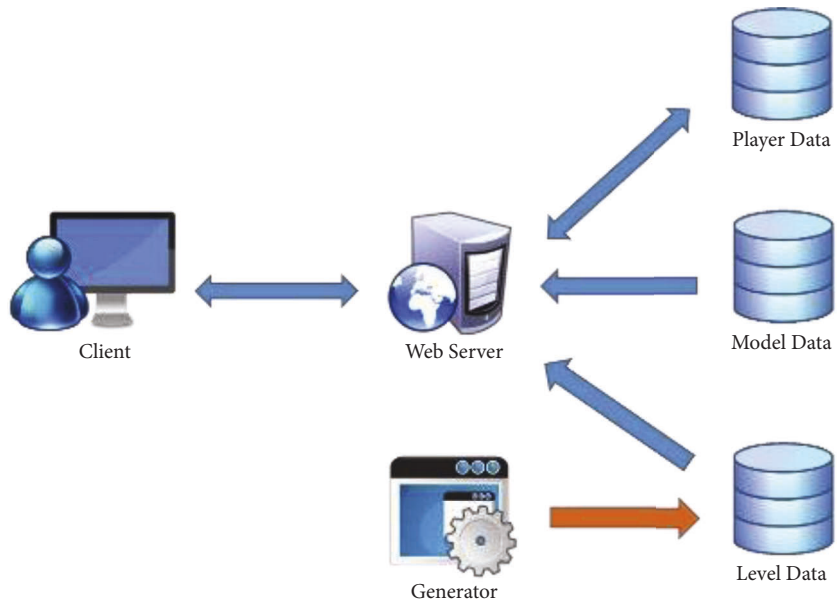


Figure 13: Animation game system architecture diagram.

provides APIs for the client to obtain the resources required by the animation game and the API for uploading player data. The animation game database mainly contains player information, models used by the level, and information about the level. Generator mainly provides automatic generation of levels. The client-side mainly provides the real operation and screen of the player's animation game, obtains the required resources, and uploads the player's data through the API provided by the web server.

In the server and animation game database part, it sets XAMPP as a cross-platform and includes a package of MySQL, Apache web server, and PHP. The environment is set up in CPU: I5-44403, 10 GHz, memory: 20 G, hard disk: 1TB, OS: Windows 10, 64 Bit. This research part of the animation game server uses PHP language to write the animation game interface, which includes animation game login, get animation game theme information, get level data, upload player records, etc. functions. The animation game database mainly stores player information and the models used in the level, and the level information is divided into the following three parts:

(1) Player information: player's animation game account, level clearance record, player's animation game time, the number of replays, the player's control buttons, and the movement of the apple.

(2) Models used in the level: use the Prefab format provided by Unity to package the models used in the animation game (apple, floor, apple bag, trap ball), and the database will store relevant information such as the placement path of the corresponding model.

(3) Level information: store relevant information about each theme level, such as the name of the theme, whether the theme is activated, and the name of the theme file used. The theme file stores the information of the level components.

Table 5: Probability setting table in the experiment.

Object name	Ground	Middle	Top
Ground	0.01	0.89	0.1
Candy	0	0.8	0.2
car1	0	0	1
car2	0	1	0
Bag	1	0	0

The client uses the unity platform and JavaScript language to implement the animation game. Its operation process mainly includes the following four parts:

(1) Log in to the animation game: the animation game will first log in to the animation game account. If it is the first time to play, the system will automatically register a new account based on the MIME code (unique code) of the device.

(2) Load theme information: when the login is completed, the system will automatically load the theme information to form a menu, and players can choose different level themes for the animation game.

(3) Load the level information and the models required by the level: when the player enters a theme to play, the system and the server will obtain the level information file of the theme and the models required by the level and initialize it according to the level information

(4) Record: during the animation game, the system will record the player's operation behavior and clearance data such as animation game time, replay times, operations, and apple movement. These data will be automatically uploaded to the animation game data when passing the level in the library.

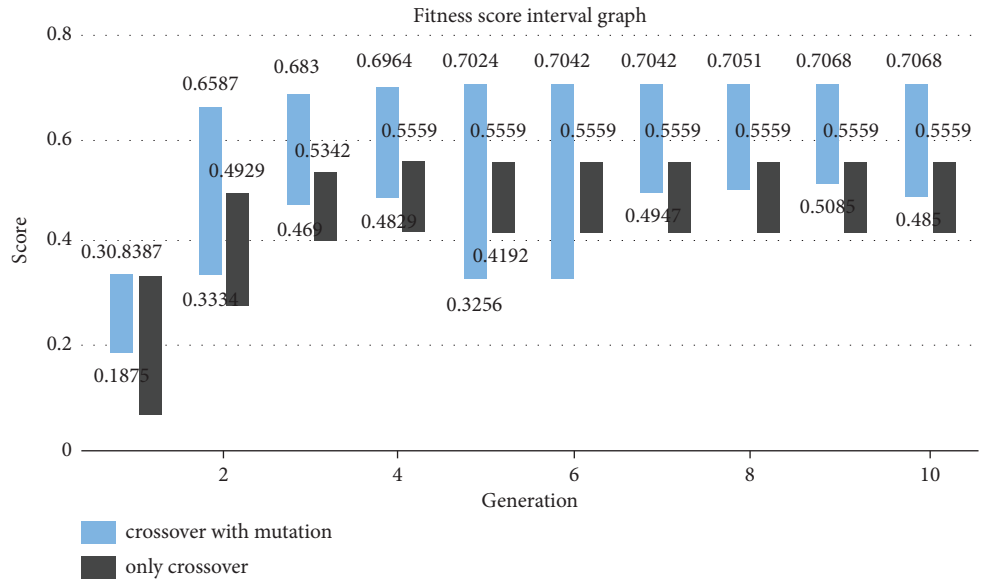


Figure 14: Fitness score change interval graph (10 generations).

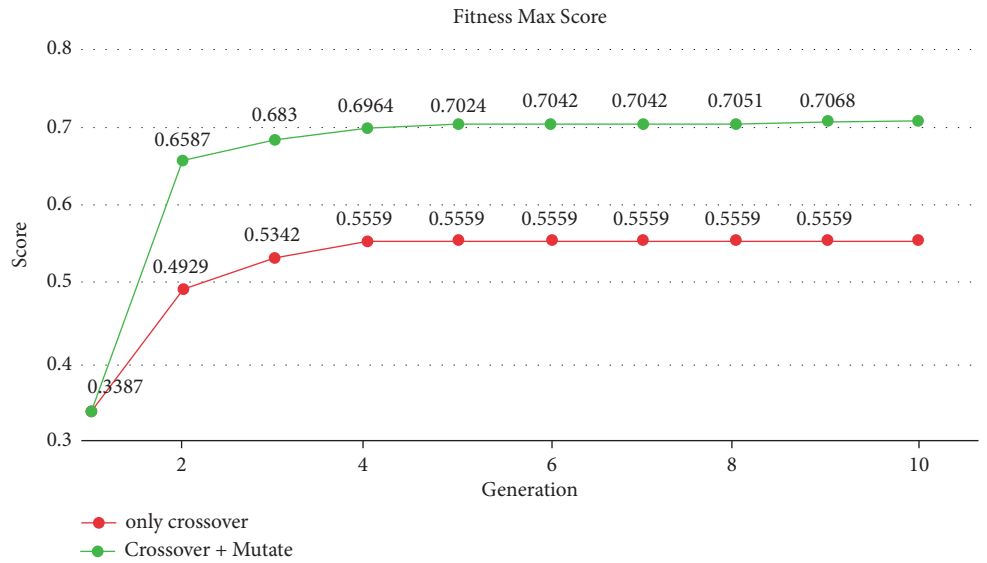


Figure 15: Fitness maximum score change line chart (10 generations).

4.2. Experimental Results. First, use custom-set parameters to generate levels through the system's genetic algorithm, and finally display the experimental results through graphs. Secondly, this research lists the average time spent by the system in each stage of execution. This result will also affect the result of whether it can be operated in real time. Finally, several completed levels will be generated, and they will be verified manually by means of manual inspection to ensure the feasibility of completeness. The experimental parameters are set as in Table 5

4.2.1. Level Generation Verification. This research uses the parameter settings in the previous section to perform genetic calculations for 10 generations and uses mutation to watch the fitness score changes of the cards in each generation (Figure 14). You can see the scores that use mutation. It is much higher because there are many turning points in the clearance path after passing through mutation. Each turning point also means the joint between one floor and the next. This research uses this feature to obtain levels with more floors.

In Figure 15, it can be seen that the genetic algorithm of this research converged rapidly after the first generation. It was because of the correction of mutation, the turning point of the clearance path, and the growth of the clearance path, which greatly improved the fitness score, but due to the limited space of the level scene, the location of the element will affect the maximum value of the turning point and the length of its clearance path.

In Figure 16 shows the standard deviation of the fitness scores of the cards in each generation. The red line segment is the result of not using mutations. It can be seen that the

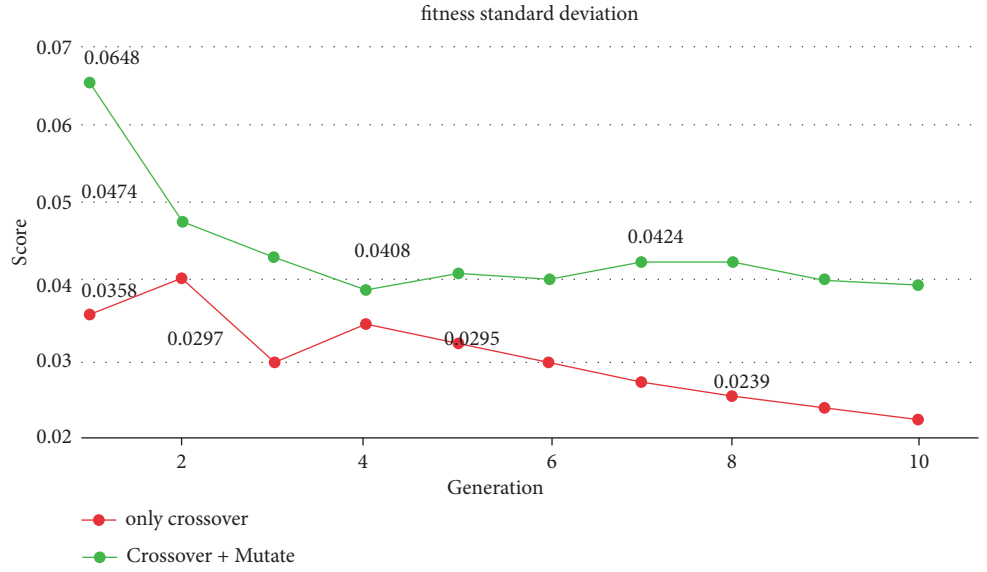


Figure 16: Fitness score standard deviation line chart (10 generations).

Table 6: Average system running time (100 generations).

Process	Average spent time (s)
Init population	17.06
Evaluate population	0.18
Crossover	2.89
Mutation	9.93
Evaluate mating pool	0.30
Elitism	0.33
Total	30.73

standard deviations have a gradual decrease, which means that the scores are getting closer and closer. The reason is that crossover easily affects the length of the gene and indirectly affects the total length of its customs clearance path and the number of turning points, which drives the overall fitness score to gradually increase, and the length only increases but does not decrease in the mating process, which makes its standard. The difference gradually decreases. After the mutation function is added to the green line in Figure 16, it can be seen that the standard deviation will gradually decrease in the previous generations. This is because the mutation of this system has a corrective effect to remedy some low-scoring levels and make them one of the levels. The scores in between can be pulled in quickly, and in the end, the standard deviation will fluctuate within a certain range due to a sudden change in probability.

4.2.2. System Running Time. Here, it allows the system to evolve for 100 generations and records the operating time of the system in each stage. Finally, it averages these data, as shown in Table 6.

It can be seen from Table 6 that the system spends the most time in the process of mating pool initialization and mutation. In the mutation process, the calculation time increases due to the correction function, but the adjustment function can also be seen in the previous section, making its genetic algorithm converge quickly. However, in the mating pool initialization, because the system does not have any genes that can mate with each other at the beginning, the system will determine the placement position according to the probability table and randomly place the animation game objects in the relative position to generate the level and the generated level. It is also necessary to obtain the clearance path through the clearance path search function, and the genetic algorithm can only be executed after the clearance path is obtained. Therefore, the operation process in this part is more complicated and therefore increases the time cost. The calculation time of crossover is affected by modifying Population_Elitism in its system configuration file. Parameter: the system will obtain the permutation and combination whose Population_Elitism parameter takes two in the crossover stage to mate one by one to obtain the mating result and mutation, but because the permutation and combination will cause the number of results to be factorial multiplied, so in the setting of this parameter, it should not be set too high; otherwise it may take up a lot of memory.

Figure 4 shows the total time spent by the system in 100 generations. The line chart shows that the time spent in the system is not proportional to the generations, but the average time spent drifting. The main reason is that the crossover in the system is mating. The top 10 high-scoring genes are selected in the pool for mating, and the mating generation is mutated. Therefore, even if more and more levels pass the threshold in the mating pool, only the first 10 levels will participate in the evolution of the next generation. Process: the main purpose is to use the highest-scored gene to mate a longer gene combination, thereby increasing the fitness score. Of course, the number of mating mothers can be changed by modifying the system parameter settings, but this also means that the execution time of crossover and mutation will increase, leading to increased time costs.

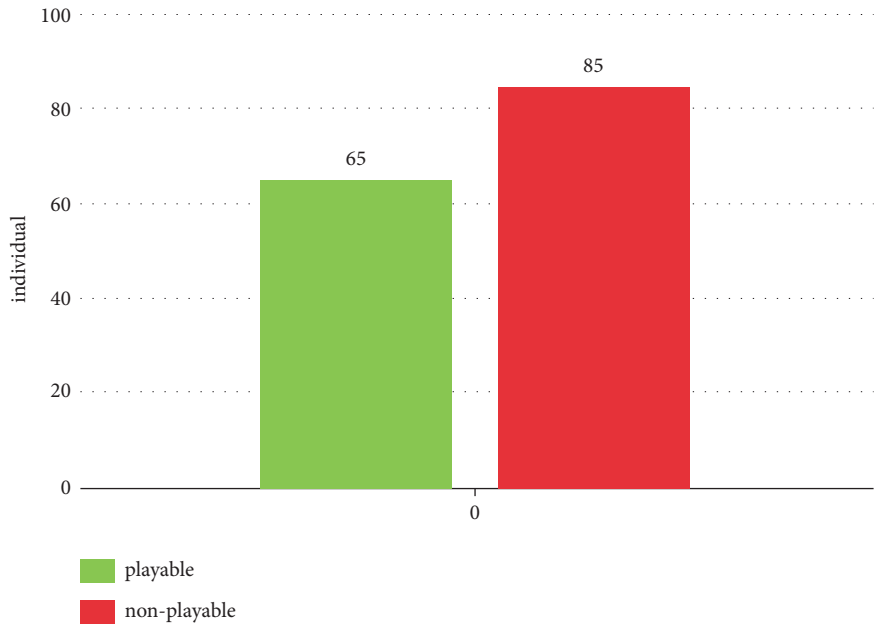


Figure 17: Total playability after the level is complete.

4.2.3. Checkpoint Complete Verification. This research uses genetic algorithms to calculate the results of 10 generations. Each generation takes the level with the highest score of 3, a total of 30 levels. These levels are completed one by one, and each level produces 5 complete levels. Finally, play these 150 levels manually to verify the playability of the completed level. The results are shown in Figure 17. The results show that there are still many levels that cannot be played after the level is completed. The main reason is that even if the system guarantees a smooth path to the level, it puts more animation game objects in the process of completion and fills up the sloped floor. With more horizontal floors, this actually indirectly interferes with the integrity of its clearance path, but if it is not completed, although more than 90% of the playable levels will be playable, there will be too many blank parts in the level. This leads to monotonous levels, so if you want to improve the current situation, you can only define more or more rigorous correction conditions, so the number of playable levels will have room for improvement.

4.2.4. Sloping Floor Correction. Through the provided equation and fixing its single-joint slope to a 30-degree slope, after the double-joint slope is a 45-degree slope, the joint problem between the sloped floor and the horizontal floor can be solved. The research shows the output results of its single-joint and double-joint floor.

5. Conclusion

5.1. Conclusion. This research used unity development tools and JavaScript language to actually create automatic generation levels of animation game systems that can load files for custom levels. The contribution of this article mainly proposes using a genetic algorithm to automatically generate levels and provide a complete level system to process the generated levels, so that the level can have more elements and improve the animation game of the level. This research method is based on the principle of ensuring that the customs clearance path is unobstructed. The research believes that as long as the customs clearance path is unobstructed, it means that the level is playable. In the genetic mutation part, this research has added some corrective mutations to make the next generation more ideal and complete. The system has also formulated many rules to make the level more perfect, and the principle of the path of clearance also allows the complete system to be better played [48].

5.2. Limitations and Future Works. Although this research has completed the generation of levels through genetic algorithms and successfully achieved a more ideal level through the level integration system, it still has limitations and future works are following:

(1) Method improvement: the currently defined fitness function is calculated for the path of the customs clearance, but in fact, the random level generated by the initialization in the entire system actually allows other animation game objects to be placed, but this research only uses the information of the customs clearance path in the end. It does not make full use of all the information possessed by initialization, so there should be a better use space here.

(2) Online real-time generation of levels: if the system can generate levels in real time, it may be necessary to set up a level database and complete the generation of complete levels by obtaining the existing levels in the database. This can achieve real-time effects.

(3) Enhancement of level integrity system: if its playability should reach more than 80%, the optimization system still sets rules for the placement of different animation game objects, but as animation game objects increase.

(4) Sloping floor: since the grid is used to represent a level in the system, there will eventually be joint problems on the sloped floor. At present, this research fixes the slope of the slope and then uses the calculated equation to calculate the correct placement position to solve the joint problem. But this research believes that the most perfect solution should support any sloped floor, so there should be more changes in a level generation.

Data Availability

The raw data supporting the conclusions of this article will be made available by the author.

Consent

The voluntary agreement is free of coercion and undue influence on research participation.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Algwiki, "What ALG is retrieved 6 29, 2016, from procedural content generation Wiki," 2014, https://ALG.wikidot.com/what-ALG-is.

[2] J. Roberts and K. Chen, "Learning-based procedural content generation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 1, pp. 88–101, 2015.

[3] A. Angry Birds, "Retrieved from Angry Birds," 2017, https://www.angrybirds.com/.

[4] H. Pei Breivold, "Towards factories of the future: migration of industrial legacy automation systems in the cloud computing and Internet-of-things context," *Enterprise Information Systems*, vol. 14, no. 4, pp. 542–562, 2020.

[5] J. Chen and K. Tai, *Pattern-Based Automatic Game Level Generation and Difficulty Evaluation*, National Dong Hwa University, Taiwan, 2016.

[6] H. Cheng, L. Wu, R. Li, F. Huang, C. Tu, and Z. Yu, "Data recovery in wireless sensor networks based on attribute correlation and extremely randomized trees," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 245–259, 2021.

[7] Y. Cheng, H. Jiang, F. Wang et al., "Using high-Bandwidth networks Efficiently for Fast graph computation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1170–1183, 2019.

[8] S. Dahlskog and J. Togelius, "A multi-level level generator," in *Proceedings of the Computational Intelligence and Games (CIG) 2016 Conference on Computational Intelligence and Games*, pp. 1–8, IEEE, Dortmund, Germany, August 2014.

[9] Y. Dai, S. Wang, X. Chen, C. Xu, and W. Guo, "Generative adversarial networks based on Wasserstein distance for knowledge graph embeddings," *Knowledge-Based Systems*, vol. 190, Article ID 105165, 2020.

[10] Diablo, "Retrieved from Diablo III Official Game site," 2017, https://tw.battle.net/d3/zh/.

[11] A. Doull, "Ascii Dreams: the Death of the level designer: procedural content generation in Games. Retrieved from the Death of the level designer: procedural content generation in Games - Part One," 2008, https://roguelikedeveloper.blogspot.tw/2008/01/death-of-level-designer-procedural.html.

[12] L. Ferreira and C. Toledo, "A search-based approach for generating Angry Birds levels. Computational intelligence and Games (CIG)," in *Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, IEEE, Dortmund, Germany, August 2014.

[13] Y. G. Fu, H. Y. Huang, Y. Guan, Y. M. Wang, W. Liu, and W. J. Fang, "EBRB cascade classifier for imbalanced data via rule weight updating," *Knowledge-Based Systems*, vol. 223, Article ID 107010, 2021b.

[14] Y. G. Fu, J. F. Ye, Z. F. Yin, L. J. Chen, Y. M. Wang, and G. G. Liu, "Construction of EBRB classifier for imbalanced data based on Fuzzy C-Means clustering," *Knowledge-Based Systems*, vol. 234, Article ID 107590, 2021a.

[15] Y. G. Fu, J. H. Zhuang, Y. ., P. Chen, L. K. Guo, and Y. M. Wang, "A framework for optimizing extended belief rule base systems with improved Ball trees," *Knowledge-Based Systems*, vol. 210, Article ID 106484, 2020.

[16] Genetic algorithm, "Genetic algorithm retrieved from Wiki," 2017, https://en.wikipedia.org/wiki/Genetic_algorithm.

[17] A. Haleem, M. Javaid, R. P. Singh, S. Rab, and R. Suman, "Hyperautomation for the enhancement of automation in industries," *Sensors International*, vol. 2, Article ID 100124, 2021.

[18] B. Horn, S. Dahlskog, N. Shaker, G. Smith, and J. Togelius, "A Comparative evaluation of procedural level Generators in the Mario AI framework," in *Proceedings of the International Conference on the Foundations of Digital Games*, pp. 1–8, Ft Lauderdale, Florida, U.S.A, April 2014.

[19] X. Y. Li, W. Lin, X. Liu, C. K. Lin, K. J. Pai, and J. M. Chang, "Completely Independent Spanning trees on BCCC data center networks with an application to Fault-Tolerant routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1939–1952, 2022.

[20] G. Liu, X. Chen, R. Zhou, S. Xu, Y. C. Chen, and G. Chen, "Social learning discrete particle Swarm optimization based two-stage X-routing for IC design under intelligent edge computing architecture," *Applied Soft Computing*, vol. 104, Article ID 107215, 2021.

[21] G. Liu, Z. Chen, Z. Zhuang, W. Guo, and G. Chen, "A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT," *Soft Computing*, vol. 24, no. 6, pp. 3943–3961, 2020a.

[22] G. Liu, X. Zhang, W. Guo et al., "Timing-aware layer Assignment for advanced process technologies Considering via pillars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1957–1970, 2022.

[23] G. Liu, W. Zhu, S. Xu, Z. Zhuang, Y. C. Chen, and G. Chen, "Efficient VLSI routing algorithm employing novel discrete PSO and multi-stage transformation," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–16, 2020b.

[24] G. Liu, Y. Zhu, S. Xu, Y. C. Chen, and H. Tang, "PSO-based power-Driven X-routing algorithm in Semiconductor design

for predictive intelligence of IoT applications," *Applied Soft Computing*, vol. 114, Article ID 108114, 2022b.

[25] N. Liu, J. S. Pan, C. Sun, and S. C. Chu, "An efficient surrogate-assisted quasi-affine transformation evolutionary algorithm for expensive optimization problems," *Knowledge-Based Systems*, vol. 209, Article ID 106418, 2020c.

[26] G. Van Den Bergen, *Collision Detection in Interactive 3D Environments*, CRC Press, Boca Raton, 2003.

[27] A. K. Inkulu, M. R. Bahubalendruni, A. Dara, and K. SankaranarayanaSamy, "Challenges and opportunities in human robot collaboration context of Industry 4.0-a state of the art review," *Industrial Robot: The International Journal of Robotics Research and Application*, vol. 49, no. 2, pp. 226–239, 2021.

[28] V. S. S. V. Prasad, M. Hymavathi, C. Rao, and M. A. Bahubalendruni, "A novel computative strategic planning projections algorithm (CSPPA) to generate oblique directional interference matrix for different applications in computer-aided design," *Computers in Industry*, vol. 141, Article ID 103703, 2022.

[29] G. A. Kumar, M. Bahubalendruni, V. Vara Prasad, D. Ashok, and K. Sankaranarayanasamy, "A novel Geometric feasibility method to perform assembly sequence planning through oblique orientations," *Engineering Science and Technology, an International Journal*, vol. 26, Article ID 100994, 2022.

[30] Z. Lu, G. Liu, and S. Wang, "Sparse neighbor constrained co-clustering via category consistency learning," *Knowledge-Based Systems*, vol. 201-202, Article ID 105987, 2020.

[31] Pacman, "Retrieved from pacman," 2017, https://www.gametop.com/download-free/pacman/.

[32] S. Shen, Y. Yang, and X. Liu, "Toward data privacy preservation with ciphertext update and key rotation for IoT," *Concurrency and Computation: Practice and Experience*, Article ID e6729, 2021.

[33] L. Lattanzi, R. Raffaeli, M. Peruzzini, and M. Pellicciari, "Digital twin for smart manufacturing: a review of concepts towards a practical industrial implementation," *International Journal of Computer Integrated Manufacturing*, vol. 34, no. 6, pp. 567–597, 2021.

[34] S. Snodgrass and S. Ontañón, "Experiments in map generation using Markov super Mario Bros (2017) retrieved from super Mario Bros," 2014, https://www.supermariobrosx.org/.

[35] J. Togelius, G. Yannakakis, O. Stanley, and C. Browne, "Search-based procedural content generation," in *Proceedings of the 2010 international conference on Applications of Evolutionary Computation, Volume Part I*, pp. 141–150, Springer, Berlin, Heidelberg, April 2010.

[36] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation:A Taxonomy and Survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, pp. 172–186, 2011.

[37] L. Ttito and J. Huacho, "Support vector machine for the implementation of the fitness function of genetic algorithms," in *Proceedings of the 2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 23–26, IEEE, Chaves, Portugal, June 2021.

[38] X. Wu, C. H. Chu, Y. Wang, and W. Yan, "A genetic algorithm for cellular manufacturing design and layout," *European Journal of Operational Research*, vol. 181, no. 1, pp. 156–167, 2007.

[39] A. Kumar Gulivindala, M. Raju Bahubalendruni, R. Chandrasekar, E. Ahmed, M. Haider Abidi, and A. Al-Ahmari, "Automated disassembly sequence prediction for industry 4.0 using enhanced genetic algorithm," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2531–2548, 2021.

[40] B. Vajgel, P. L. P. Correa, T. Tossoli De Sousa et al., "Development of intelligent Robotic process automation: a utility Case study in Brazil," *IEEE Access*, vol. 9, pp. 71222–71235, 2021.

[41] S. Wang, Z. Wang, K. L. Lim, G. Xiao, and W. Guo, "Seeded random walk for multi-view semi-supervised classification," *Knowledge-Based Systems*, vol. 222, Article ID 107016, 2021.

[42] J. Wewerka and M. Reichert, "Robotic process automation - a systematic mapping study and classification framework," *Enterprise Information Systems*, pp. 1–38, 2021.

[43] Z. Yu, X. Zheng, F. Huang, W. Guo, L. Sun, and Z. Yu, "A framework based on sparse representation model for time series prediction in smart city," *Frontiers of Computer Science*, vol. 15, no. 1, pp. 151305–151313, 2021.

[44] Y. Zhang, G. Huang, X. Liu, W. Zhang, H. Mei, and S. Yang, "Refactoring android Java code for on-demand computation offloading," in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 233–248, ACM, October 2012.

[45] H. Zhang, J. L. Li, X. M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Generation Computer Systems*, vol. 122, pp. 130–143, 2021a.

[46] Y. Zhang, Z. Lu, and S. Wang, "Unsupervised feature selection via transformed auto-encoder," *Knowledge-Based Systems*, vol. 215, Article ID 106748, 2021.

[47] X. Zheng, C. Rong, and T. Badarch, "Foreword to the special issue of green cloud computing: Methodology and practice," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 23, Article ID e5425, 2019.

[48] W. Zou, L. Guo, P. Huang, G. Lin, and H. Mei, "Linear time algorithm for computing min-max movement of sink-based mobile sensors for line barrier coverage," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 2, Article ID e6175, 2022.