Research Article

An Effective Multimedia Item Shell Design for Individualized Education: The Crome Project

Irene Cheng¹, ² and Anup Basu¹

¹ Department of Computing Science, University of Alberta, Canada T6B 2E8
² Department of Computer and Information Science, University of Pennsylvania, PA 16802, USA

Correspondence should be addressed to Irene Cheng, chenglin@seas.upenn.edu

Received 10 August 2007; Revised 15 January 2008; Accepted 12 February 2008

Recommended by S. Takamura

There are several advantages to creating multimedia item types and applying computer-based adaptive testing in education. First is the capability to motivate learning by making the learners feel more engaged and in an interactive environment. Second is a better concept representation, which is not possible in conventional multiple-choice tests. Third is the advantage of individualized curriculum design, rather than a curriculum designed for an average student. Fourth is a good choice of the next question, associated with the appropriate difficulty level based on a student’s response to the current question. However, many issues need to be addressed when achieving these goals, including: (a) the large number of item types required to represent the current multiple-choice questions in multimedia formats, (b) the criterion used to determine the difficulty level of a multimedia question item, and (c) the methodology applied to the question selection process for individual students. In this paper, we propose a multimedia item shell design that not only reduces the number of item types required, but also computes difficulty level of an item automatically.

The concept of question seed is introduced to make content creation more cost-effective. The proposed item shell framework facilitates efficient communication between user responses at the client, and the scoring agents integrated with a student ability assessor at the server. We also describe approaches for automatically estimating difficulty level of questions, and discuss preliminary evaluation of multimedia item types by students.

Copyright © 2008 I. Cheng and A. Basu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Along with content creation, multimedia has potential for use in both knowledge acquisition and innovative testing. Rather than traditional paper-and-pencil formats, audio, video, graphics, and animation are being conceived as alternative means for more effective learning and testing in the future [1–11]. One advantage of computer-based learning and testing is to advance education to individualization; instead of handing out the same set of learning or testing material to the entire class, the next tutorial or exam question can be adaptively selected based on the current performance of an individual student. Since the learning curve of every student is different, individualization is an effective approach for educators and teachers, who can monitor a student’s progress better than before using informative and summative computer generated information, and can provide immediate assistance if necessary. The challenge of adopting such an approach is the large collection of tutorials and questions in the database, which needs to cater to all learning levels, and has to be available on demand. Differing from traditional multiple-choice item formats that can easily accommodate different questions, multimedia items require specific screen layouts depending on a question’s contents and the type of media used. The programming and development cost may outweigh the benefits of individualization if each question requires a new design and implementation. In a distance-learning environment, the question content can easily overload the server-client capability if the transmitted data is not properly designed and regulated. Another advantage of computer-based learning is to make designed and organized curriculum centrally available, through wired or wireless networks, to remote locations where resources are insufficient to prepare educational materials locally.

In this paper, we propose a novel approach for designing multimedia item shells to address the issues of scalability, reusability, and portability. Our design takes into
The rest of the paper is organized as follows: terminology used in this work is described in Section 2. Section 3 discusses the requirements of an effective multimedia item shell. Section 4 introduces our design strategy, where some of the examples implemented are also described. Section 5 discusses the testing and scoring strategy. Section 6 describes a brief student evaluation of some of our multimedia item types. Finally, Section 7 gives the conclusion and discusses future work.

2. TERMINOLOGY

We use the following terminology in this work. 

Item—is used in the same context as question or question item.

Multiple choice (MC) item shell—is a template used to generate MC items. Multiple questions can be generated using the same shell by inputting different texts to describe the question and the choices.

Multimedia item shell—is a template used to generate items with similar characteristics. While an MC item shell has a fixed screen layout, a multimedia item shell can be mapped onto different screen layouts. We use a category code to control such mapping. The item shells implemented in our framework are multimedia item shells.

Parameters—are control variables. By assigning different values to a parameter, multiple items with different screen layouts can be generated.

3. ITEM SHELL REQUIREMENTS

Requiring special software installation prior to executing an application often turns learners away. In order to motivate and engage learners, runtime support of common browsers such as Internet Explorer and Fire Fox should be considered. Furthermore, in an adaptive testing environment, online communication with the server is required; therefore, running a testing session as an offline application is not an option. In order to design effective multimedia item shells, we also need to consider the following.

(1) Understanding the current pen-and-paper question contents and screen layout, and determining what type of multimedia content are a good match for each question.

(2) Designing a format conversion system by incorporating multimedia content so that the concepts associated with a question are better presented. The idea is to make a student feel more engaged and motivated. It is important that the format conversion does not alter the difficulty of a question; otherwise, the curriculum needs to be reorganized. For example, if a Grade 6 multiple-choice question requires a student to select the organ not located on the body, and after conversion, the new format requires the student to drag and drop the organs to the correct location of the body (Figure 1), then the question difficulty is increased because the student needs to have additional knowledge of an organ's positions.

(3) Enhancing the new format to a generic item shell with the capability of enabling or disabling embedded multimedia types by changing some control values. For example, the drag and drop item shell shown in Figure 1 can be designed for dragging 2D images and 3D objects, as well as text. An item shell can also be designed to embed audio, video, animation, still picture, and text description, so that question items of similar screen layouts requiring one or a combination of these multimedia content can share the same item shell.

(4) Extending an item shell to accommodate multiple screen layouts. Completing an electric circuit may sound very different from picking groceries for a shopping cart (Figures 2(a) and 2(b)), but both can be handled by the same item shell in our design. Note that the same item shell can also be used to present the following layouts:

(a) ordering components in a specified sequence;
(b) choosing the correct operator to complete an equation;
(c) dragging the correct descriptions to different locations of an image;
(d) spelling a word by rearranging the alphabets;
(e) mapping an image with the correct description; and
(f) classifying components into their correct categories.

(5) Many abstract concepts that cannot be captured using pen-and-paper format can easily be demonstrated using multimedia content, such as, three-dimensional display, audio, and video. Therefore, computer-based multimedia items should be more diversified than traditional items.
Put the ammeter to the place where the electrical current is 0.4 A.

Re-order the list from the city with the most people to the city with the fewest by dragging and dropping the city names in the column on the right.

Put EQUAL sign and NOT EQUAL sign to the correct positions.

Re-order the letters to make a word.

Match the national flags.

Figure 2: The capability of displaying different multimedia contents in the same item shell reduces the number of item shells to be implemented.
A cost-effective item shell should be portable, scalable, and reusable so that not only the number of item shells is minimized, but also the number of questions is minimized. Our design strategies are described in Section 4.

4. ITEM SHELL DESIGN STRATEGIES

Many learning objects available on the web are implemented using flash. One reason is the runtime support from web browsers; another reason is its appealing graphics, animations, and sound effects. However, flash-based questions are not reusable; specific design and implementation is required for a question and it is inefficient if not impossible to implement a flash item shell to satisfy requirements (3) and (4) above. Furthermore, although flash can present 2D graphics and animations nicely, its 3D capability is not comparable with Java 3D, at least in its current version. We chose Java to implement item shells because of its 2D and 3D capability, as well as the flexibility to satisfy requirements (3) and (4). To animate an object in flash, multiple frames of 2D orientations have to be defined similar to the production of traditional cartoons. Although drawing a sequence of vector-based objects (Figure 3) is fast in flash, creating an elaborate and photorealistic object for animation is time consuming. Besides, a student is not able to alter the sequence of predefined actions and navigate around the object interactively. By contrast, only a single Java 3D object needs to be created which can be manipulated in any orientation. 3D mesh formats like OBJ and OFF can be reusable by different question items, and a large number of off-the-shelf Java 3D objects are available in the public domain. Java runtime is also fully supported by web browsers, and more importantly, it is platform independent and thus portable. Effective 3D rendering can also be achieved in an OpenGL environment. However, it has to run locally as a stand-alone application, and cannot be used as an online network application for adaptive testing. Compared to earlier versions, the current JavaScript version is more powerful in capturing graphics content including drag-and-drop components, as well as highlighting text. Although they can be used as learning items and take a shorter time to render than applet items, the readable script at a client machine leads to security and copyright issues.

In order to achieve reusability and scalability, we propose using (Figure 4):

1. screen layout and item shell mapping;
2. item and item shell independence;
3. composite item shells in addition to unit item shells;
4. question seed association; and
5. parameter-based difficulty level estimation.

While question content, for example, text description, is unique per question, there are metadata (e.g., number of dragged objects in Figure 2(g) that do not need to change. By analyzing a question's content in the current curriculum, features independent of question content are extracted as parameters for designing multiple item shells. Groups of related parameters are then identified to form question seeds. Multiple question items can be generated, by assigning appropriate values to these parameters. Two fundamental parameters are the question and the answer parameter strings, which are composed of tokens. Question description and control values are extracted by parsing these strings. The question parameter controls how a question is displayed to a student, while the answer parameter guides the scoring. Although an item shell defines a group of parameters, not every parameter appears in a question. For example, one question may require 3D navigation, while the other may need video clips to be displayed. We apply a standard screen layout to mask an item shell in order to maintain a uniform and consistent appearance. Masking means activating only screen components used by the current question. For example, the 3D animation component between the image and the audio panels is masked in Figure 5 because no animation is required in this question. Our design also supports composite questions composed of subquestions. Figure 5 shows an example of a composite question. Note that a student can switch from one question to another by clicking on the tags displayed at the bottom. Composite questions are generated by item shells with multiple components. The advantage of using a composite shell is to let multiple questions share common descriptions, or to group related questions together into a logical unit.

In the current implementation, PHP is used as the web language to communicate between a database (MySQL) and Java applet-based item shells (Figure 6). Note that our item shells are portable and can be integrated with other web languages. Each student has his or her own logon session; allowing statistics, such as time stamp and IP address, to be recorded for every student. PHP is also used to communicate with the integrating engine at the server. After receiving the response string from a client applet, instead of passing the entire string to the server engine for further processing, the string is first divided into logical tokens by PHP and used to update the database. An index is then created for the database record. Only the index is passed to the engine, which uses the index to retrieve a reduced data set that is required by the marking agent. This approach ensures that the server engine is not flooded with the large amount of data.
generated by multiple student sessions running concurrently. Since each item shell has its unique design and parameter composition, it is associated with its own marking agent. A student response is scored in the range [0, 1] where 0 means wrong and 1 means correct. Partial marks, between 0 and 1, can also be given.

An important process in our framework is to record the performance history and scoring statistics allowing abnormal learning patterns of students to be detected when they occur. This enables educators to take corrective actions on time. In order to assess individual student abilities, it is necessary to select a sequence of appropriate items dictated by a student’s responses, allowing scores to converge to the student’s ability level. In order to monitor a student’s performance history and scoring statistics allowing corrective actions to be quite large, say 2, before the curve becomes steep. The characteristic of an IRT curve is that the range \([-\infty, \infty]\) of the student’s ability (\(\theta\)) levels. The 3PL model defines the probability of getting a correct response based on guessing. Changing its value will affect the lower asymptote. If the value is 0, students with limited ability have a low probability of getting a correct answer.

An IRT model can be associated with one, (1PL) two (2PL), or three (3PL) parameters. Only \(a_i\) and \(b_i\) are considered in 2PL. In 1PL, only \(b_i\) is considered. A general representation of the item response function is given below:

\[
P_i(\theta_j) = P(u_i = 1 \mid \theta_j, a_i, b_i, c_i) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta_j - b_i)}}.
\]

2PL is obtained by setting \(c_i = 0\), and 1PL (Rasch model) is obtained by setting \(c_i = 0\) and \(a_i = 1\).

The 3PL model was first proposed in [14]. Given an item \(i\) and a student \(j\), \(\theta_j\) denotes the ability of student \(j\), and \(u_i (0 \leq u_i \leq 1)\) denotes the correctness of the student’s response on item \(i\). The 3PL model defines the probability of a correct answer to item \(i\) for student \(j\) in terms of \(P_i(\theta_j)\). Traditional “true/false” and “multiple choice” (1 out of 4) test items have a probability of 50% and 25%, respectively, on guessing correctly. In contrast, the probability of guessing correctly in most of the multimedia items designed in our framework is extremely low, if not impossible (\(c_i\) approaches 0). For example, in a drag and drop item with \(M\) objects to place (Figures 2 and 12), the probability of guessing correctly is \(1/M!\). Nevertheless, we adopt the 3PL IRT model in our system in order to cover the possibility of correct guessing for some items.

In our item shell framework, we are interested in the following.

1. How to map the scores output from the marking agents to the IRT input?
2. How to define the terminating condition in order to detect the converged ability level?
Given a group of students at a particular grade, we start by assuming that every student has an average ability $\theta = 0$ in the range $[-3–3]$ and is given the first question with average difficulty $b_i = 0$. It is possible that a value outside the $[-3–3]$ range is generated resulting from the responses of a student. If such pattern persists, it is an indication that the ability level of the student is either below or above the current grade being considered. In our framework, difficulties are assigned in the range $[0–1]$. Let $\beta$ be the set of $b_i$ defined in the IRT model and let $\mathcal{D}$ be the set of difficulty values in the range of $[0–1]$ used by our marking agents. To address question (1) above, we establish a 1-to-1 mapping between $\mathcal{D}$ and $\beta$. To address question (2), we use the item information function for item $i$:

$$I_i(\theta_j) = \frac{P_i'(\theta_j)^2}{P_i(\theta_j)(1 - P_i(\theta_j))},$$  

(2)

and standard error function in IRT:

$$SE(\theta_j) = \frac{1}{\sqrt{I_i(\theta_j)}},$$  

(3)

where $P_i'(\theta_j)$ is the first derivative of $P_i(\theta_j)$.

5. TESTING AND SCORING STRATEGY

The adaptive testing process terminates and the response curve is considered to have converged when $SE(\theta_j)$ is less than a predefined threshold. Regardless of the starting difficulty level given to a student, his or her ability can be assessed in a limited number of items as illustrated by the convergence rate of the curve shown in Figure 8. Readers interested in the mathematical analysis of IRT can refer to [12–14] for details. During the adaptive testing process, the server engine returns the newly estimated ability based on
The true/false type of question is vulnerable to random guessing. If a student does not know the answer, (s)he can guess and the probability of guessing correctly is 50%. In this case, it is not possible to determine the student's true knowledge level because whichever initial difficulty level is assigned, the student’s response curve stays in equilibrium. In normal circumstances, the response curve should follow one of the following three trends [17] (an example of a response curve is given in Figure 8).

(i) If the initial knowledge level is higher than that of the student, the curve should decrease and then converge at his or her level.

(ii) If the initial level is lower than that of the student, the curve should increase and then converge.

(iii) If the initial level matches that of the student, the curve should stay more or less horizontal.

The probability of guessing correctly in multiple-choice questions with 4 choices is 25%. If the number of correct answers is used to assess a student's performance, random guessing can award a student 25% even if the student's ability is lower. Note that the point of inflection in Figure 7 divides the probability of getting or not getting a correct response into 50%. Assigning a zero value to $c_i$ will affect the probability $P_i(θ_j)$ given a student’s ability, and thus will affect $l_i(θ_j)$ and the next item to be selected. In order to assess a student's ability correctly, the 3PL IRT model taking the guessing parameter $c_i$ into account is therefore more accurate. While the guessing probabilities for true/false and multiple-choice items are obvious, it may not be possible to precisely compute the guessing parameter value for all the multimedia items due to their different screen presentations. It is therefore important to design the items in a way that minimizes the guessing probability as much as possible. At the same time, it is necessary to monitor the adaptive testing response curve to detect possible guessing patterns.

5.2. Parameters-based estimation of difficulty level for math item types

Parameter-based strategy is a more general approach for assigning initial difficulties to items. We use math questions as examples to illustrate the concept. Figure 11 shows an item
Question: distribute the weights evenly into the 4 baskets

(a) Wrong answer! one correct answer is:
Number in basket #0: 10, 20, 30
Number in basket #1: 16, 44
Number in basket #2: 60
Number in basket #3: 20, 40

(b) Figure 11: (a) An interactive math question and (b) a student answer.

Figure 12: Interface used in the evaluation experiment.

requiring a student to distribute the numbers into four bins so that the sum in each bin is the same. We define parameters \( n_{\text{bkt}} \) (number of bins) and \( n_{\text{nbr}} \) (number of objects to distribute) to control the generation of multiple items, as well as the difficulty levels of the questions generated. For example, when solving the question “distribute the numbers so that the sum in each bin is equal” (Figure 11), the difficulty level of a question is defined by the function \( f(n_{\text{bkt}}, n_{\text{nbr}}) \). The difficulty level increases as \( n_{\text{bkt}} \) or \( n_{\text{nbr}} \) increases. Additional difficulty can be introduced by using decimal instead of integer numbers. We verified the feasibility of our approach by conducting evaluation experiments.

5.3. Evaluation of the parameter-based strategy

5.3.1. Methodology

We extended the concept of IRT and used 2PL, coupled with measurement of average time taken to solve problems, to fit a linear regression model and examine the correlation between the difficulty levels generated by our strategy with the predefined difficulty levels. The calibration was done by seven students to rate the difficulty of each item based on the percentage of correct responses. 2PL was used, since it was almost impossible to guess the correct answer for the given question format; the value of parameter c was close to zero. Mathematical details will not be discussed here for brevity. However, we will describe the design of the evaluation experiment and discuss results. The user interface of the evaluation program is shown in Figure 12. The questions used for evaluating the automatic difficulty estimation algorithm are described in Table 1. The higher is the question ID, the greater is the difficulty level. Questions (1) to (6) involve summation and questions (7) through (12) involve multiplication.

The order in the table is the same as the one used in the evaluation. Maximum time (in units of 10 seconds) assigned to each question was based on a roughly predicted difficulty during the calibration process, that is, more difficult questions are allocated more time. Participants’ familiarities with the questions were not taken into account during the assignment of maximum times. A participant’s answer, time
The entire experiment follows a zero-feedback procedure.

Participants were allowed to use assisting tools such as a calculator, but not allowed to discuss with one another. The entire experiment follows a zero-feedback procedure.

A procedure to solve a PRODUCT question is:

(a) Factorize each number into prime numbers.
(b) Place the appropriate numbers into different baskets so that each basket has the same count of different prime numbers.

Participants were allowed to use assisting tools such as a calculator, but not allowed to discuss with one another. The entire experiment follows a zero-feedback procedure.

(1) A participant is introduced to the graphical user interface and the method for each type of question.
(2) The participant answers two warm-up SUM questions in order to get used to the user interface and the method for solving the question.
(3) The participant answers the set of SUM questions sequentially.
(4) The participant answers two warm-up PRODUCT questions.
(5) The participant answers the set of PRODUCT questions sequentially.

### 5.3.3. Results and analysis

Each participant’s ability was considered as his or her total mark scaled in the range between \([-3, 3]\). Depending on the estimated abilities, each question’s difficulty parameter \(b\) is calculated using IRT. Based on the experimental data (not shown here), the linear regression equation for estimating the difficulty of the SUM questions is:

\[
b = -6.44 + 0.47n_{\text{het}} + 2.77\left(\frac{n_{\text{hbr}}}{n_{\text{hbt}}}\right) - 0.74\text{ID}, \tag{4}
\]

where ID varies between 1 and 6 (Table 1) depending on the calibrated difficulties. The correlation between the calibrated and experimental values was found to be \(R^2 = 0.95\).

The linear regression equation for estimating the difficulty of the PRODUCT questions (ID between 7 to 12) is

\[
b = -14.74 + 3.52n_{\text{het}} + 2.77\left(\frac{n_{\text{hbr}}}{n_{\text{hbt}}}\right) - 1.08\text{ID} \tag{5}
\]

with \(R^2 = 0.99\). The high \(R^2\) values (close to 1.0) indicate that the difficulty parameter \(b\) estimated by our algorithm has very high correlation with the \(b\) obtained from the calibrated values. Hence the proposed parameter-based strategy for estimating difficulty level is validated.

### 6. STUDENT FEEDBACK ON MULTIMEDIA ITEM TYPES

We have received positive feedback from K-12 student groups visiting our research center regarding the appeal of multimedia item types to students. Extensive user studies with some students were conducted during August 2007. Some of the findings on four students are summarized in Table 2. The numbers in Table 2 are counts of multimedia item types used in the evaluations. Grade 11 students evaluated 33 items in total, whereas the Grade 7 student evaluated 36 items. For example, out of 33 multimedia items, student 1 was satisfied with 31, neutral on 1, and dissatisfied with 1. Student 2 took less (more) time to work with 14 (11) of the 33 items, compared to their corresponding pen-and-paper versions; and took about the same time with both forms of tests for 8 items.

Note that the four students in Table 2 had somewhat different backgrounds. Three were in Grade 11 and one was in Grade 7. Among the Grade 11 students, students 2 and 3 had taken computer-programming courses, while student 1 did not have any programming knowledge. It can be seen that these students in general were both satisfied with the multimedia item types and also preferred computer-based testing. However, there were some differences in the evaluations: (a) students 2 and 3 had very similar evaluation...
Table 2: Summary of detailed evaluations by some students.

<table>
<thead>
<tr>
<th>Feedback</th>
<th>Student1 Grade 11</th>
<th>Student2 Grade 11</th>
<th>Student3 Grade 11</th>
<th>Student4 Grade 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfaction</td>
<td>Satisfied</td>
<td>Neutral</td>
<td>Dissatisfied</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Preference</td>
<td>Computer-based</td>
<td>Pen and paper</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time taken</td>
<td>Less computer</td>
<td>Less pen and paper</td>
<td>About the same</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>18</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>11</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td>Not recorded</td>
<td>Recorded</td>
<td>Properly</td>
<td></td>
</tr>
<tr>
<td>Preference</td>
<td>Computer-based</td>
<td>Pen and paper</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Summary of detailed evaluations by some students.

and timing results since they were both from the same grade and with good programming and user-interface knowledge, these skills may have given them an edge in performing the computer-based tests quite fast; (b) student 1, though very interested in computer-based multimedia item types, was relatively slower in working with the computer test interfaces, and in most cases performed the pen-and-paper tests faster; (c) student 4, though satisfied and interested in the multimedia item types, was unable to record precise time data properly. This may be a result of the slight immaturity of a Grade 7 student compared to Grade 11 students. In future evaluations with junior students, it is necessary to find appropriate means of accurately recording the time taken on pen-and-paper tests without involving a costly monitoring process.

7. CONCLUSIONS AND FUTURE WORK

In this paper, the important prerequisites for effective item shell design for individualized education were discussed, along with implemented examples. We addressed the reusability, portability, and scalability issues by introducing the concept of item seeds and parameters, which are used to control automatic generation of difficulty levels and scoring.

Experiments on using IRT for estimating difficulty and timing results since they were both from the same grade and with good programming and user-interface knowledge, these skills may have given them an edge in performing the computer-based tests quite fast; (b) student 1, though very interested in computer-based multimedia item types, was relatively slower in working with the computer test interfaces, and in most cases performed the pen-and-paper tests faster; (c) student 4, though satisfied and interested in the multimedia item types, was unable to record precise time data properly. This may be a result of the slight immaturity of a Grade 7 student compared to Grade 11 students. In future evaluations with junior students, it is necessary to find appropriate means of accurately recording the time taken on pen-and-paper tests without involving a costly monitoring process.

ACKNOWLEDGMENTS

The implementation support of the CROME project team is gratefully acknowledged. This project is funded by grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada; Informatics Circle of Research Excellence (iCORE), Alberta; and Castle Rock Research Corp., Canada. Parts of this paper were presented at a special session in IEEE Int. Conf. on Multimedia and Expo, Beijing, 2007.

REFERENCES

Submit your manuscripts at
http://www.hindawi.com