The Design and Evaluation of a Java-Based Software Tool for Teaching Page Replacement Algorithms

SUKANYA SURANAUWARAT
Graduate School of Applied Statistics
National Institute of Development Administration
118 Seri Thai Rd., Bangkapi, Bangkok 10240
THAILAND
sukanya@as.nida.ac.th http://as.nida.ac.th/~sukanya

Abstract: - Page replacement algorithms are one of the most important aspects of a page-based virtual memory system. In this paper, a tool implemented as a Java application and designed as an aid to the study of page replacement algorithms is presented. This tool uses graphical animation to convey the concepts of various page replacement algorithms including Optimal, First-In-First-Out (FIFO), Least Recently Used (LRU), Most Recently Used (MRU), Clock, Enhanced Clock, Least Frequently Used (LFU), and Most Frequently Used (MFU) replacement algorithms. The tool is unique in a number of respects. First, it differentiates the read-access pages from the write-access ones, since the cost of replacing a page that has been modified is greater than for one that has not. Second, it allows the user to practice and test his understanding of the concepts he has learnt through a very easy-to-use graphical user interface. Third, it allows the user to compare the performance of two different algorithms or that of the same algorithm with different conditions in an easy manner. The tool has been used in an operating system course and has demonstrated effectiveness in assisting student learning in a statistically significant way.

Key-Words: - Educational Software, Animation Tool, Computer Science Education, Page Replacement Algorithms, Virtual Memory, and Operating System

1 Introduction

Most modern general-purpose operating systems use paging for virtual memory management. In a page-based system, the virtual address space of each process is divided up into equal-sized chunks called pages and can be assigned to the corresponding units in physical memory called frames or page frames. When a process references a page that is mapped into its address space but not loaded into physical memory, a page fault is said to occur and the process is suspended until the missing page is brought into memory. However, if all the page frames are in use, a page replacement algorithm must then decide which page currently in memory will be replaced. Page replacement algorithms are one of the most important memory management policies in a page-based virtual memory system [1]. Many page replacement algorithms have been developed over the years which have as their objective that the page to be replaced should be the page least likely to be referenced in the near future. Therefore, an important measure of goodness for a page replacement algorithm is the number of page faults generated for a particular reference string (i.e., a sequence of page numbers referenced by a process during its execution) and number of frames. The better the algorithm is the lower the number of page faults that are generated.

As an aid to the study of page replacement algorithms, the author has developed an interactive Java-based simulator that uses graphical animation to convey the concepts of various page replacement algorithms. The simulator is unique in a number of respects. First, in addition to the number of page faults, the simulator also evaluates the performance of a page replacement algorithm in terms of the cost of replacing pages, since the cost of replacing a page that has been modified is greater than for one that has not. This is because the former must be written back out to disk before the new page can be read in. The second unique feature of the simulator is that it allows the user to practice and test his understanding of the concepts he has learnt through a very easy-to-use graphical user interface. The third unique feature of the simulator is that it allows the user to compare the performance of two different algorithms or that of the same algorithm with different conditions. By using this feature of the simulator, the user can explore under what conditions a page replacement algorithm performs well and under what conditions it is better than another algorithm.

The simulator was used in an operating systems course at the author's institute and its impact on student learning was evaluated. Evaluating the data using t test to compare means indicated that using the simulator does increase student performance in a statistically significant way.

The remainder of this paper is organized as follows: section 2 is a brief overview of the page replacement algorithms used in the simulator, section 3 gives a description of the simulator, section 4 discusses the evaluation results of the simulator, section 5 discusses related work, and section 6 draws some conclusions.

2 Overview

There are various page replacement algorithms. The simulator uses the algorithms listed below (which are discussed in [2][3][4][5]).

- **Optimal**: replaces the page that will either never be needed again, or will not be used for the longest period of time. This "unrealizable" algorithm is usually used only as a benchmark by which other algorithms can be judged.
- **First-In-First-Out (FIFO)**: replaces the page that has been in memory the longest. FIFO incurs low overhead but generally does not predict future page usage accurately.
- **Least Recently Used (LRU)**: replaces the page in memory that has not been referenced for the longest time. LRU generally predicts future page usage well but incurs significant overhead.
- **Most Recently Used (MRU)**: replaces the page in memory that has most recently been used.
- **Clock** (or **Second Chance**): is a modified form of the FIFO algorithm. It treats the page table as a circular list of pages, and uses the reference bit associated with each entry in the page table and a pointer (the "clock hand") into the page table. The reference bit is set whenever a page is referenced. When a page must be replaced, the algorithm checks the page table entry pointed to by the clock hand. If the referenced bit for that page is set, it is cleared and the clock hand is advanced. It continues in this manner until it finds an entry whose reference bit is off, and in that case it selects that page for replacement.
- **Enhanced Clock** (or **Third Chance**): makes the Clock algorithm more powerful by increasing the number of bits that it employs. That is, it makes page replacement decisions using two bits: the reference and modify bits. Whenever a page is referenced, the reference bit is set; whenever modified, the modify bit is set. When a page must be replaced, the algorithm begins with the page table entry pointed to by the clock hand. If the reference and modify bits for that page are set, the reference bit is cleared and the clock hand is advanced; if the reference bit is set but the modify bit is not, the reference bit is cleared and the clock hand is advanced; if the reference bit is clear but the modify bit is set, the modify bit is cleared (and the algorithm notes that the page must be copied out before being replaced) and the clock hand is advanced; if both the reference and modify bits are clear, the page in that frame is replaced.
- **Least Frequently Used (LFU)**: selects a page for replacement if the page has not been used often in the past. This algorithm keeps a counter of the number of references that have been made to each page. Pages with the lowest counts are replaced while pages with higher counts remain in primary memory.
- **Most Frequently Used (MFU)**: replaces the page with the largest count. This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

3 Description of the Simulator

The simulator is written using Java 6. It has three operating modes: simulation, practice, and comparison modes. In simulation mode, the user can watch virtually step-by-step how an algorithm works or watch it run straight through from the beginning until the end. In practice mode, the user can decide which page replacement algorithm he wants to test his understanding of, and then he can try to select the pages that will be replaced by himself through a very easy-to-use graphical user interface; after that he can check whether his answers are correct or not with the simulator. In comparison mode, the user can compare the performance of two different algorithms or that of the same algorithm with different number of frames in an easy manner. Each mode is described below.

3.1 Simulation Mode

Fig. 1 shows a snapshot of the simulator during a simulation in simulation mode. The top panel of the screen allows the user to control the simulator as he

desires; specifically, the user can select which algorithm to animate through a drop-down list box located at the top of the panel. When the user selects the algorithm, the user can also specify his own reference string and the number of page frames, or he can use the default values.

The reference string can be specified by the user by typing the string into the reference string text box or alternatively, the user can have one randomly generated for him by a single click on the "Random" button. If a more specific random reference string is desired, then the user needs to double-click on the "Random" button. This will cause a pop-up window to appear. Through this window, the user can input values for the possible length of the reference string and the possible page numbers that can be in the reference string. By default, the pages in the reference string are read accesses. The user can specify that the pages are write accesses by typing "*" next to the page numbers. The simulator also allows the user to save any reference strings for later use by clicking on the "Save this reference string" checkbox. Otherwise, the reference string will be lost when the user exits the program. Above the "Random" button is the "Concept" button, when this button is clicked, a pop-up window appears with a description of the currently selected algorithm.

Along the bottom of the top panel, there are buttons that allow the user to control the animation. The user can start and stop the animation whenever he wishes by clicking on the "Start" and "Stop" buttons. Alternatively, the user can choose to watch the algorithm step-by-step by repeatedly clicking the "Next" button. Also, the speed of the animation can be changed using the slider.

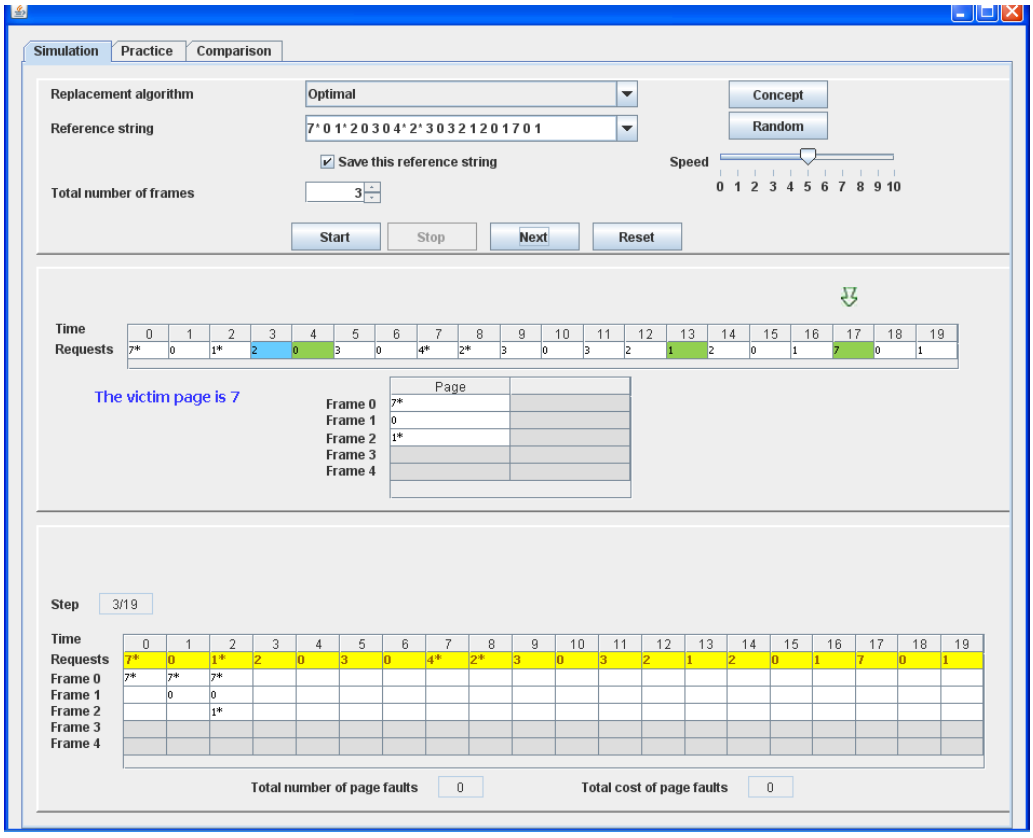

Fig. 1: A snapshot of the simulator during a simulation using the Optimal algorithm

In Fig. 1, the state of the simulator at virtual time 3 is shown during a simulation using the Optimal algorithm with the reference string of: '7* 0 1* 2 0 3 0 4* 2* 3 0 3 2 1 2 0 1 7 0 1' and the number of frames is set to 3. The *virtual time* of a process represents the progress the process makes as it executes. In other words, the virtual time is advanced each time the process makes a memory reference. The simulator uses the blue highlight to indicate which page is currently referenced. As shown in the middle panel of the screen, page 2 is referenced at virtual time 3. Since page 2 is referenced for the first time and all of the frames – three frames, in this example – are occupied, it is necessary to replace one of the current three frame pages, the victim page, with page 2. When this

happens, the message "A page fault has occurred" followed by a second message "Selecting a victim page" are displayed. Then the animation shows how a victim page is selected to be replaced under the Optimal page replacement algorithm. That is, it scans the rest of the reference string starting from the page referenced during the next virtual time (i.e., virtual time 4) until it determines which page is referenced latest or never; that one is selected as the victim page. The scanning process is shown by a moving arrow above the reference string. During the scan, it found that pages 0, 1, and 7 are referenced again at virtual time 4, 13, and 17, respectively, as shown by the green highlights. After finding the next reference for each page, it ends the scanning process at virtual time 17 concluding that page 7 is the victim page. Once the victim page is determined, the frame in the middle panel that has that page will be highlighted in red and will blink a couple times to draw the user's attention. After that, the simulator will replace the victim page with the new page and then display the message "The page fault cost is 2". At this point, the table in the bottom panel will be updated with the data from virtual time 3. Then virtual time will increment to virtual time 4. Note that the page fault cost is calculated based on whether the victim page had been modified or not. If the victim page is modified, the total cost of a page fault is 2 I/O transfers, while the cost of an unmodified victim page is 1 I/O transfer. The cost of replacing a page that has been modified is greater than for one that has not, because the former must be written back out to disk before the new page can be read in.

The main feature of the bottom panel is a table. The table has a row for the request and a row for each frame, as well as, a column for each virtual time *t*. A table entry at Frame *i* and column *j* shows the page loaded at time *t* in Frame *i* after r_j (which is the number of the page referenced by the process at its virtual time *j*) has been referenced. If the entry is highlighted with a different color, the page shown in the entry was loaded as a result of the missing page fault. Each column heading is a virtual time; under the headings, there are pages from the reference string. Below the table, the total number of page faults and the total cost of page faults that have occurred so far are displayed. Note that, only page faults occurring after the frame allocation is initially filled are counted.

The next algorithm to be discussed is the LRU algorithm. When the LRU algorithm gets to virtual time 3 with the same reference string that was used with the Optimal algorithm shown in Fig. 1, instead of looking ahead the algorithm looks back to determine which page will be the victim page. This is animated by the simulator in the same way as described for the Optimal algorithm, except it highlights the prior occurrence of each page before the current virtual time instead of the future occurrence of the page. Once the least recently used page is determined, it becomes the victim page which is replaced by the current reference string page. After that, the table in the bottom panel will be updated with the data from virtual time 3. Then virtual time will increment to virtual time 4.

The MRU algorithm is also animated by the simulator in a similar way to the LRU algorithm except that it highlights only the most recently occurrence of the page prior to the current virtual time.

The rest of the algorithms, which are the Enhanced Clock, the Clock, the FIFO, the LFU, and the MFU algorithms, are very similar in their simulation. For the Enhanced Clock algorithm, the page in each frame is associated with a reference bit (R bit) and a modify bit (M bit); and the pointer (arrow) is set to the oldest page, as shown Fig. 2. When the Enhanced Clock algorithm gets to virtual time 3 with the same reference string that was used with the Optimal algorithm shown in Fig. 1, a page fault occurs. It then scans R and M bits of the page in each frame to find a page that has both R and M bits set to zero (i.e., R and M bits are off); this is simulated with a moving highlighted region as in Fig. 3. During the scan, each time it encounters a page that does not have both R and M bits set to zero, it turns off one of these two bits according to the rule described in section 2 and continues on. Once it encounters the oldest page with both R and M bits set to zero – page 0 in this example – the frame in the middle panel that has that page will be highlighted in red and will blink a couple times to draw the user's attention. After that, the simulator will replace the victim page with the new page following the same process described early for the Optimal algorithm. Note that the numbers displayed after "/" in the table at the bottom panel of Figs. 2 and 3 represent the reference or modify bit value of each page and the symbol "<" in the same table represents the current position of the pointer.

The Clock algorithm is also animated by the simulator in a similar way to the Enhanced Clock algorithm except that it uses single-bit reference information to make page replacement decisions. Hence, the second column of the table in the middle panel displays only the R bit of each page.

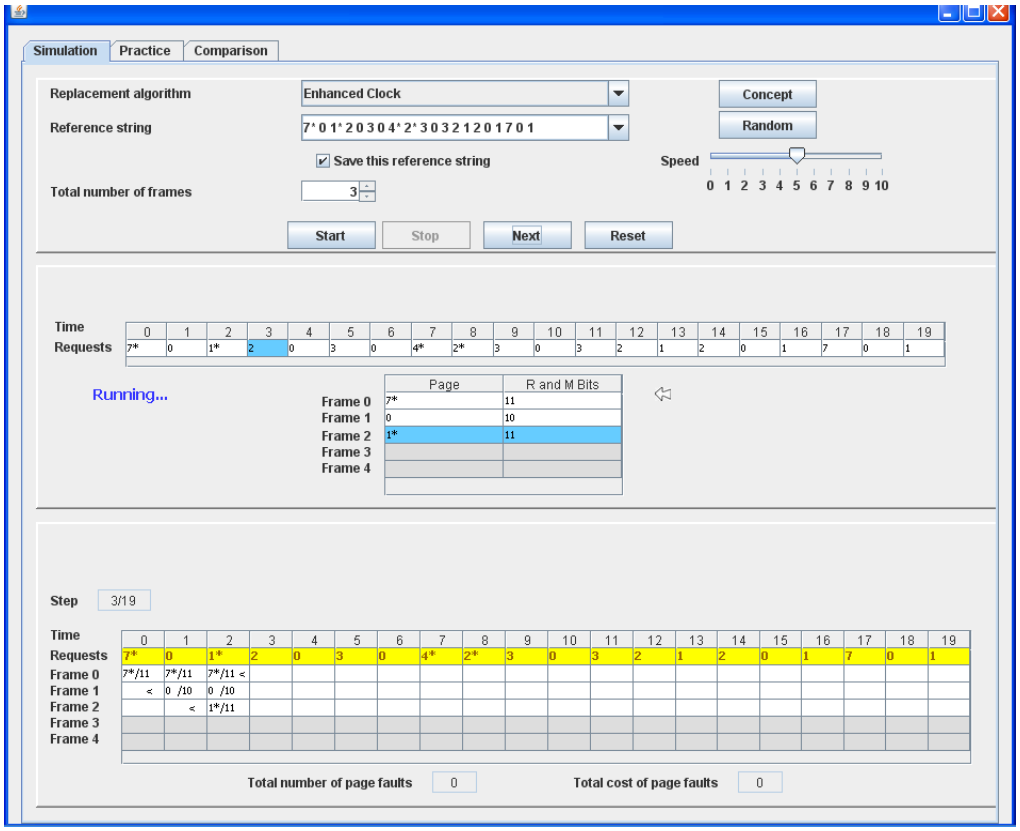

Fig. 2: A snapshot of the simulator during a simulation using the Enhanced Clock algorithm before a page fault

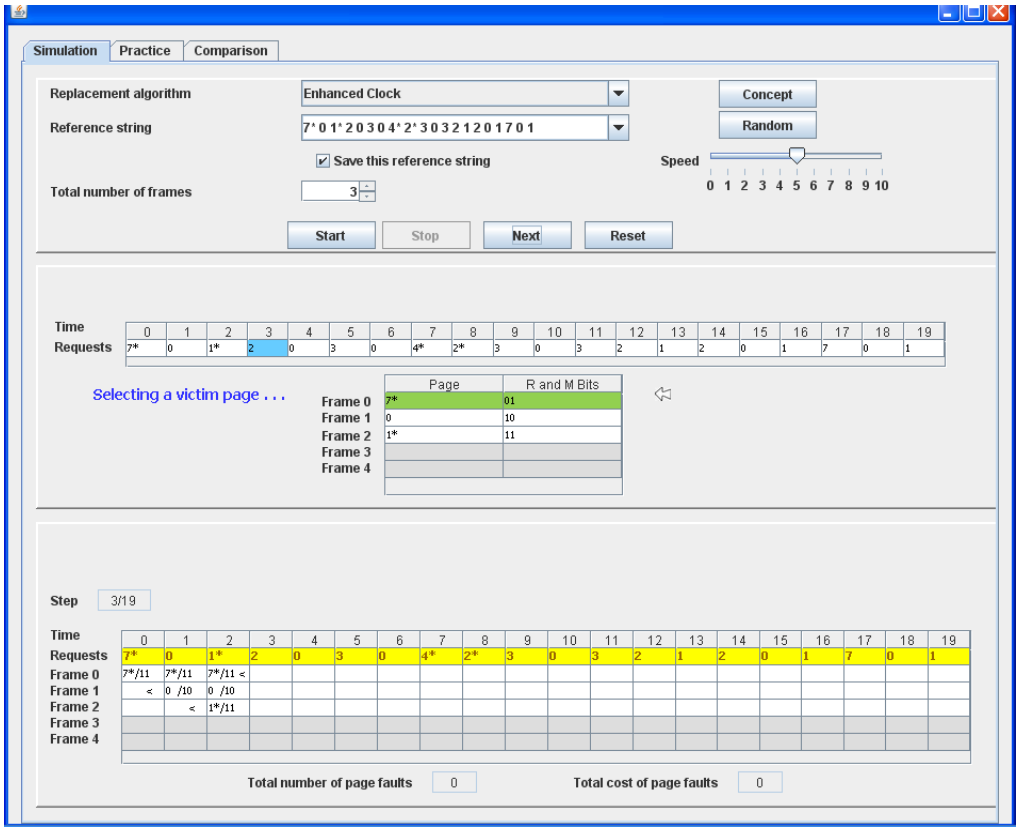

Fig. 3: A snapshot of the simulator during a simulation using the Enhanced Clock algorithm after a page fault

The rest of the algorithms, the FIFO, the LFU and the MFU algorithm, run in a similar way to the Enhanced Clock algorithm except that there is no pointer set to the oldest page and the information in the second column of the table is different. In the case of the FIFO algorithm, the second column is the time each page is loaded and in the case of the other two algorithms, LFU and MFU, the second column of the table is a counter (i.e., a counter of the number of references that have been made to each page). When a page fault occurs, the FIFO algorithm will scan for the page with the oldest loaded time; whereas the LFU and the MFU algorithm will scan for the page with the smallest and the largest counts, respectively.

3.2 Practice Mode

In Figs. 4 through 7, different snapshots of the simulator in practice mode are shown. As in the simulation mode, the user needs to specify which page replacement algorithm will be used, as well as a reference string, and the number of page frames; otherwise, the default values will be used. In practice mode, there are two ways of practicing: One-step-at-a-time and All-at-once. By clicking on the corresponding radio button located under the "Random" button, the user can choose which way to practice. Both ways of practicing are described in detail below.

Fig. 4 shows a snapshot at virtual time 3; at this point, the simulator is waiting for the user input about virtual time 3. The graphic in the middle indicates the user input for virtual time 2 was correct; if the user input was incorrect then the graphic would say it was wrong as well as highlighting the incorrect answers in the right table. Note that a correct answer for the Clock algorithm includes the reference bit value, and for the Enhanced Clock algorithm includes both reference and modify bit values. Also, for the sake of convenience, the user does not need to type the "*" that is used to indicate that the pages are write accesses. After indicating if the answer is correct the data in the left table is updated to reflect the correct data. In the table at the bottom of the panel the current virtual time is highlighted in green and the page faults are highlighted in red. In Fig. 5, the user has inputted data for virtual time 3 in the left table and clicked the "Submit" button; the simulator has acknowledged the answers are correct and the data in the tables is updated.

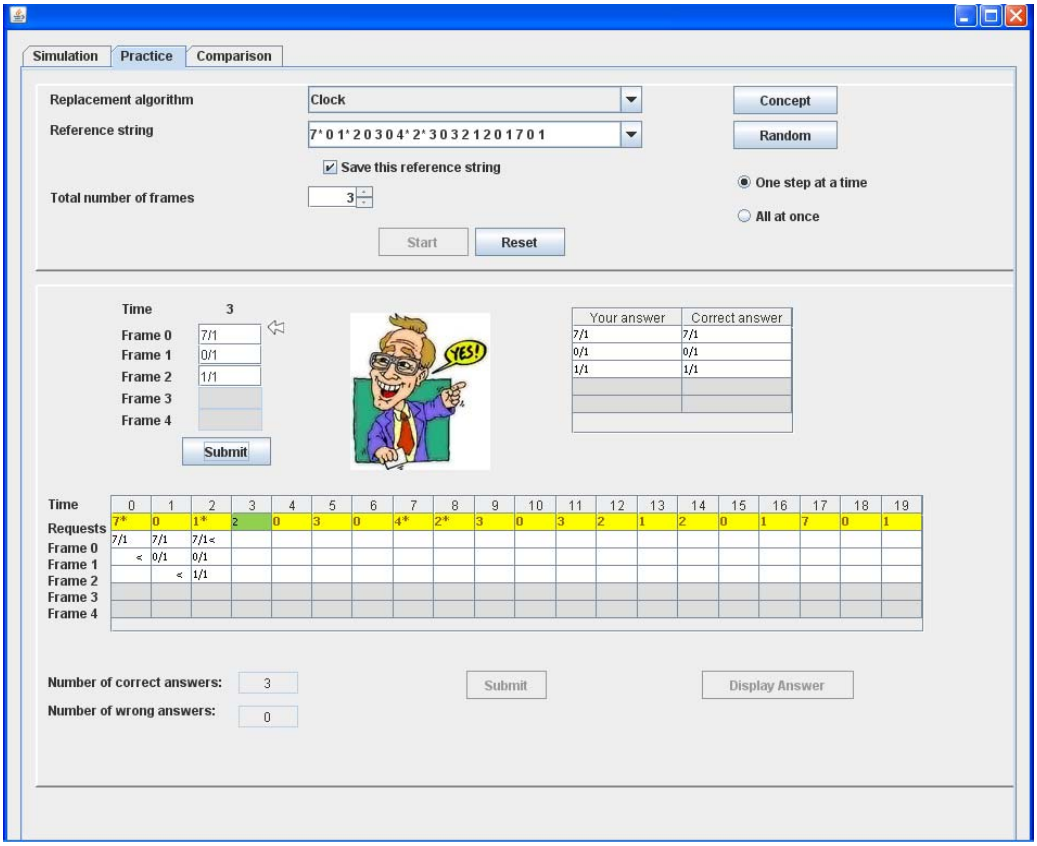

Fig. 4: A snapshot of the simulator at virtual time 3 in One-step-at-a-time practice mode

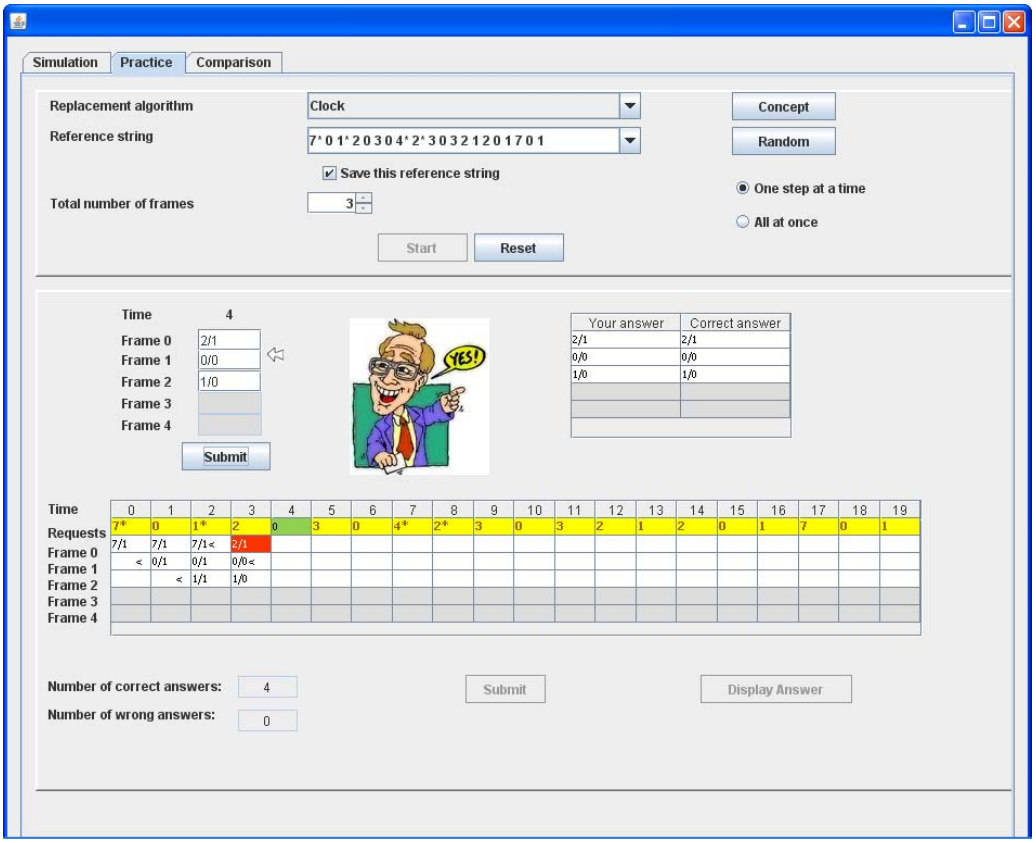

Fig. 5: A snapshot of the simulator at virtual time 4 in One-step-at-a-time practice mode

Fig. 6 is a screen shot of the simulator in All-at-once practice mode. In All-at-once practice, the user is expected to enter all the data into the practice table at the bottom for all virtual time. Once the user has input as much data as desired, clicking the "Submit" button will cause the simulator to highlight any incorrect answers and enable the "Display Answer" button. When the user clicks the "Display Answer" button, the correct answers will be displayed in the practice table as shown in Fig. 7.

3.3 Comparison Mode

A page replacement algorithm can be evaluated by running it on a particular reference string, and computing the number of page faults and the page fault cost. The fewer the number of page faults and the lower the page fault cost, the better the performance is, and the better the replacement algorithm is. In comparison mode, the user can compare the performance of two different algorithms. The user can also compare the performance of the same algorithm with different number of page frames as shown in Fig. 8, which is an example of Belady's anomaly, the phenomenon where adding more page frames does not reduce the number of page faults. Moreover, the user can use this mode to explore the effect of a pattern of page accesses on different algorithms. For example, the user can explore which pattern of page accesses for which MRU performs better than LRU, or vice versus.

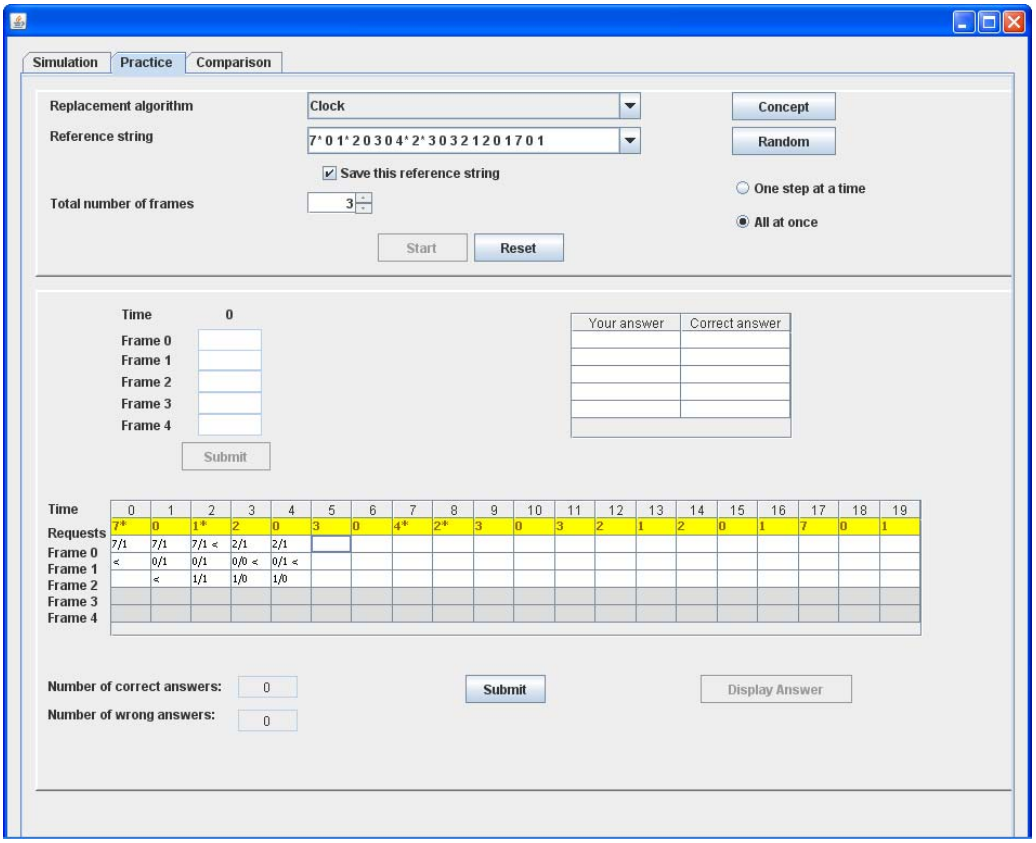

Fig. 6: A snapshot of the simulator in All-at-once practice mode

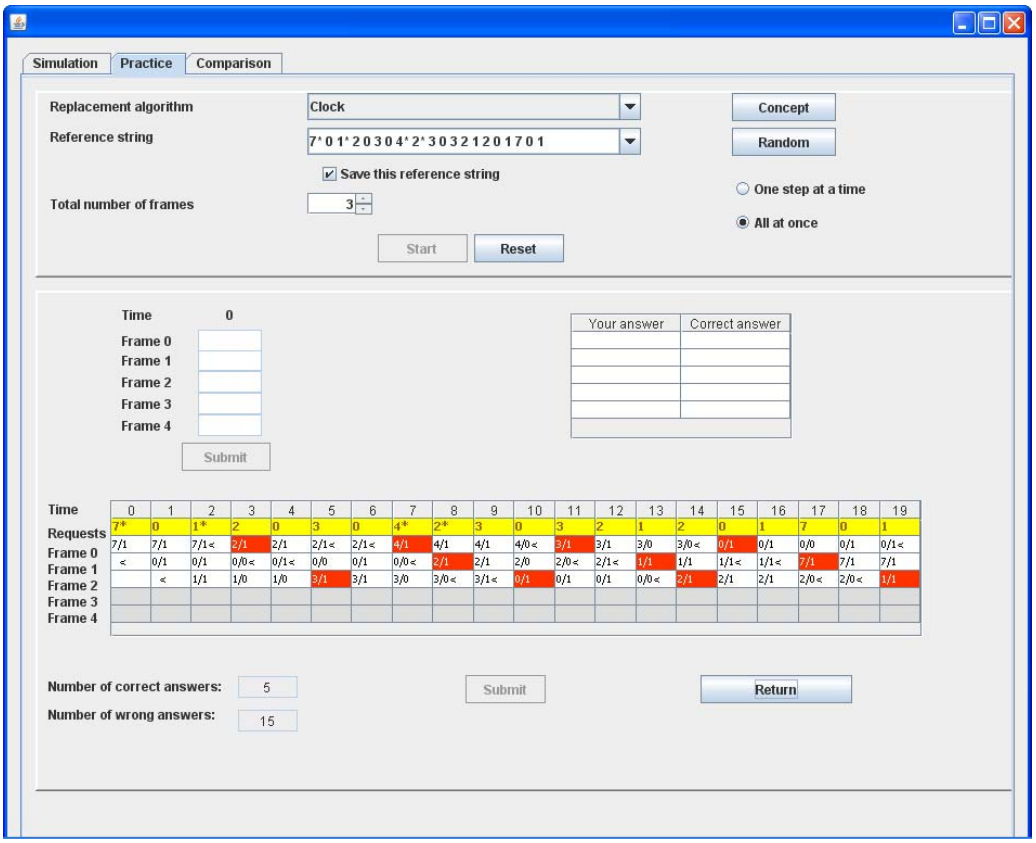

Fig. 7: A snapshot of the simulator in All-at-once practice mode after the "Display Answer" button is clicked

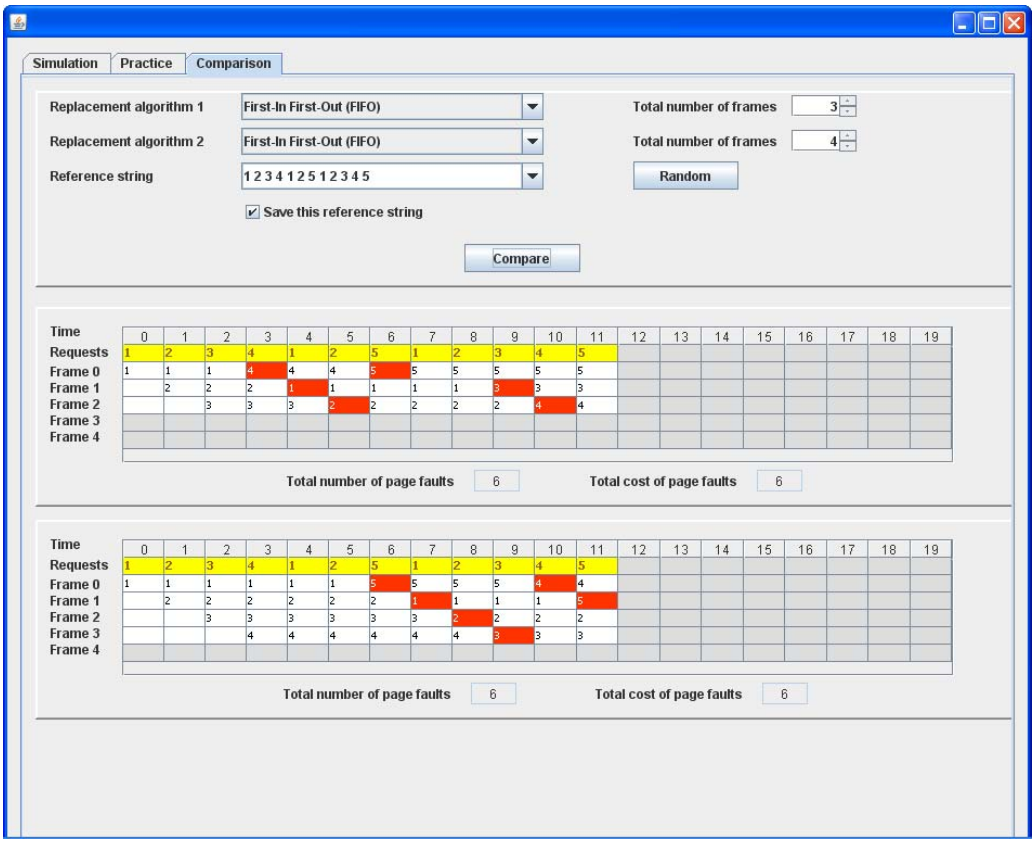

Fig. 8: A snapshot of the simulator in comparison mode

4 Evaluation

The true usefulness of the simulator designed as a supplement to an operating systems course is in its effectiveness in aiding students in learning the concepts of page replacement algorithms. The simulator was used in an operating systems course in the fall of 2009. The sample for the effectiveness study consisted of twenty-two students enrolled in the operating systems course. Prior to the study students' prior knowledge of page replacement algorithms was tested. All students scored zero for prior knowledge so each student was randomly assigned to one of three groups. The first group (group 1) learned page replacement only from the text (N = 8). The second group (group 2) learned page replacement through using the simulation mode of the simulator and reading the text (N = 7). The third group (group 3) learned page replacement through using the simulation and the practice modes of the simulator and through reading the text (N = 7). The students in groups 2 and 3 received written instructions about how to use the simulator, but they were not forced to view the animation of any particular set of algorithm, reference string, and number of page frames. Rather, they were allowed to interact with the animation in any manner they desired in order to avoid unfairly favoring the animation groups. The null hypothesis is that each group will perform the same. In other words, using the simulator will not affect student performance.

Fig. 9 shows a sample question used in the pretest. The questions in the posttest were different but had the same structure as the one shown in Fig. 9.

Consider the following reference string of pages required for a process:

1, 2, 4, 3, 7, 1, 3, 4, 5, 7, 2, 3, 4, 6

How many page faults would occur using the Optimal page replacement algorithm and assuming four page frames? Count initial loads as page faults.

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Requests	1	2	4	3	7	1	3	4	5	7	2	3	4	6
Frame 0														
Frame 1														
Frame 2														
Frame 3														

Total number of page faults = _____

Fig. 9: A sample pretest question

The results are shown in Fig. 10 and are displayed in three boxplots. Even before calculating p-values it is apparent that groups 2 and 3 did much better than group 1 and that group 3 also outperformed group 2.

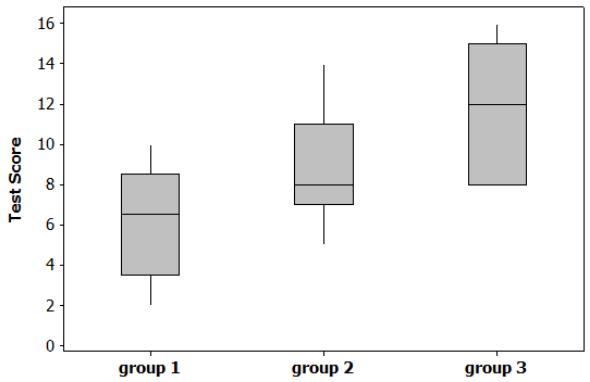

Fig. 10: Boxplots of the three groups of student scores

The result when group 1 and group 2 were compared using a t test was that group 2 learned the content better than group 1 ($t(12.4) = -1.946$, $p = 0.037$). This shows (at $\alpha = 0.05$ level) that group 1 did not learn as well as group 2 and thus that students who use the simulator even just in simulation mode performed better than students who only read about page replacement.

The result when group 1 and group 3 were compared using a t test was that group 3 learned the content better than group 1 ($t(11.9) = -3.776$, $p = 0.0013$). This shows (at $\alpha = 0.05$ level) that group 1 did not learn as well as group 3 and thus that students who use the page replacement software in simulation and practice modes performed much better than students who only read about page replacement.

The result when group 2 and group 3 were compared using a t test was that group 3 learned the content better than group 2 ($t(11.9) = -1.821$, $p = 0.047$). This shows (at $\alpha = 0.05$ level) that group 2 did not learn as well as group 3 and thus that students who use the page replacement software in both simulation and practice modes performed better than students who only used simulation mode.

The results show that using the simulator increases student performance especially when students use both simulation and practice modes. One possible reason is that students retain more content when they are actively engage in learning activities that use the content knowledge. As a growing body of literature shows, learners who are actively engaged with visualization technology have consistently outperformed learners who passively view graphics [6][7][8]. For example, Byrne et al. [6] conducted an experiment in which viewers were forced to make predictions about what they would see. These viewers scored significantly better on a posttest than others who merely watched identical animation without predicting what they would watch.

The students' opinions on the simulator were collected. Some comments from the students are: "The simulator is easy to use"; "I understand the concept better when using the simulator", and "The simulator is more appealing than the textbook". Overall, the simulator is very satisfactory as a learning aid to the students.

Finally, former students' performance on the topic of page replacement algorithms in their comprehensive exams that were took place in 2011 show that students who had access to the simulator (i.e., students who enrolled in the operating systems course in the fall of 2009) scored significantly better than the rest of the students. To be more specific, the means of this group of students is 27.21 while the mean of the rest of the students is 16.16.

5 Related Work

In the past two decades, a number of visualization and animation tools have been developed and used in many areas of computer science and engineering education [9]-[16]. While the achievement of learning outcomes as a result of using visualizations and animations has been mixed, there is evidence indicating that carefully designed visualizations and animations can have beneficial learning effects. For example, engagement of the learners attention [6][7][8] and the ability to control the pace of the visualization [17] appear to be key factors in building effective visualization and animation tools.

Not many visualization and animation tools have been developed for learning page replacement algorithms. Several animations were developed and used by English and Rainwater [18] as part of their lecture in an operating systems course. Among these animations, three of them are used in teaching Optimal, FIFO, and LRU page replacement algorithms. While these animations can enhance the teacher's lecture, they do not teach the concepts of the algorithms. They simply show the window sliding left to right to illustrate the contents of the current pages in memory; if a page fault occurs, a large red X is placed beneath the windows for that page request. The animations are also accessible through the web [19] for anyone to use. However, they do not allow the users to interact with them that much; only one data set is used, and the same animation plays over and over again.

PAGE [20] is an interactive visualization tool that demonstrates how each algorithm works based

on how the operating system implements the algorithm. For example, PAGE visualizes FIFO algorithm as a queue, with the oldest page at the head of the queue and places the most recent arrival at the tail of the queue. When a page must be replaced, PAGE removes the page at the head of the queue, and adds the new page at the tail of the queue. Another example is the Clock algorithm which is visualized as a circular list of pages in memory, with the clock hand pointing to the oldest page in the list. When a page fault occurs and no empty frames exist, then the R bit is inspected at the hand's location. If R is 0, the new page is put in place of the page the clock hand points to; otherwise, the R bit is cleared. Then, the clock hand is incremented and the process is repeated until a page is replaced. This tool supports FIFO, Clock, and LRU page replacement algorithms and its variant versions. Although the user can specify his own reference string, the number of page frames, which is visualized as the length of the queue in the case of FIFO or as the length of the circular list in the case of Clock, is fixed. To fully benefit from this tool, users should read the textbook written by Tanenbaum [21], since this tool was designed to be closely aligned with its content.

Considering the existing tools, neither provides any function that is similar to the practice and the comparison modes of the author's simulator. Also, they do not distinguish whether the pages in the reference string are read or write accesses, so they do not calculate the page fault cost and do not support the Enhanced Clock algorithm, which uses information about pages being referenced or modified in order to make page replacement decisions.

5 Conclusion

This paper presents an interactive Java-based simulator that demonstrates the concepts of various page replacement algorithms through animation. There are three operating modes for the simulator; the first is simulation mode, the second is practice mode, and the third is comparison mode. In simulation mode, the user can start and stop the simulation whenever he wishes, and watch the simulation straight through from the beginning until the end, or watch it step-by-step. In practice mode, the user can test his understanding in two ways: One-step-at-a-time and All-at-once. In comparison mode, the user can experience Belady's anomaly and can explore under what conditions a page replacement algorithm is better than another algorithm.

The simulator was used in an operating systems course in the fall of 2009. Pretest and posttest were given to three groups of students – the group (group 1) that learned page replacement only from the text, the group (group 2) that learned page replacement from the text and through using the simulation mode of the simulator, and the group (group 3) that learned page replacement from the text and through using the simulation and the practice modes of the simulator. The evaluation results show that groups 2 and 3 did much better than group 1 and that group 3 also outperformed group 2. Therefore, the simulator is effective in aiding students in learning page replacement algorithms, especially when its practice mode is used, since this mode of the simulator requires the users to be actively engaged with it more.

The simulator is available to anyone who requests it.

Future work will include developing new animation tools for learning other operating system concepts

References:

[1] P. J. Denning, "Virtual Memory", *Computing Surveys*, Vol. 2, No. 3, 1970, pp. 153-189.

[2] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts*, 8th ed., John Wiley & Sons, 2010.

[3] W. Stallings, *Operating Systems: Internals and Design Principles*, 7th ed., Prentice Hall, 2012.

[4] G. Nutt, *Operating Systems*, 3rd ed., Addison Wesley, 2004.

[5] L. F. Bic and A. C. Shaw, *Operating Systems Principles*, 1st ed., Prentice Hall, 2003.

[6] M. Byrne, R. Catrambone, and J. Stasko, "Evaluating Animations as Student Aids in Learning Computer Algorithms", *Computers & Education*, Vol. 33, No. 4, 1999, pp. 253-278.

[7] C. Hundhausen, S. Douglas, and J. Stasko, "A Meta-Study of Algorithm Visualization Effectiveness", *Journal of Visual Languages and Computing*, Vol. 13, No. 3, 2002, pp. 259-290.

[8] S. Grissom, M. McNally, and T. Naps, "Algorithm Visualization in CS Education: Comparing Levels of Student Engagement", *Proceedings of the 2003 ACM Symposium on Software Visualization*, 2003, pp. 87-94.

[9] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards, "Algorithm Visualization: The State of the

Field", *ACM Transactions on Computing Education*, Vol. 10, No. 3, Article 9, 2010.

[10] W. S. Gilley, "Animations and Interactive Material for Improving the Effectiveness of Learning the Fundamentals of Computer Science, Master's Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, 2001.

[11] D. Schweitzer and W. Brown, "Using Visualization to Teach Security", *Journal of Computing Sciences in Colleges*, Vol. 24, No. 5, 2009, pp. 143-150.

[12] S. H. Rodger, E. Wiebe, K. M. Lee, C. Morgan, K. Omar, and J. Su, "Increasing Engagement in Automata Theory with JFLAP", *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education*, 2009, pp. 403-407.

[13] P. Bauer, J. Leuchter, V. Steklý, "Simulation and Animation of Power Electronics in Modern Education", *Proceedings of the 4th WSEAS International Conference on Applications of Electrical Engineering*, 2005, pp. 48-52.

[14] P. Marambeas, P. Stergiopoulos, S. Papathanasiou, P. Bauer, and S. Manias, "Interactive Multimedia Material for an Electrical Power Quality Course", *WSEAS Transactions on Advances in Engineering Education*, Issue 7, Vol. 4, 2007, pp. 141-146.

[15] M. G. Sánchez-Torrubia, M. A. Sastre-Rosa, V. Giménez-Martínez, C. Escribano-Iglesias, "Visualization on Learning Mathematics Concepts for Engineering Education", *Proceedings of the 4th WSEAS / IASME International Conference on Engineering Education*, 2007, pp. 232-235.

[16] M. G. Sánchez-Torrubia, C. Torres–Blanc, and S. Krishnankutty, "Mamdani's Fuzzy Inference eMathTeacher: a Tutorial for Active Learning", *WSEAS Transactions on Computers*, Issue 5, Vol. 7, 2008, pp. 363-374.

[17] P. Saraiya, C. Shaffer, D. Mccrickard, and C. North, "Effective Features of Algorithm Visualizations", *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 2004, pp. 382-386.

[18] B. M. English and S. B. Rainwater, "The Effectiveness of Animations in an Undergraduate Operating Systems Course", *Journal of Computing Sciences in Colleges*, Vol. 26, No. 5, 2006, pp. 53-59.

[19] COSC 3355 Animations, http://cs.uttyler.edu/Faculty/Rainwater/COSC3355/Animations/index.htm

[20] S. Khuri and H. Hsu, "Visualizing the CPU Scheduler and Page Replacement Algorithms", *Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education,* 1999, pp. 227-231.

[21] A. Tanenbaum, *Modern Operating Systems*, 3rd ed., Prentice Hall, 2009.