Abstract: - As an aid to the study of page replacement algorithms that are used in the context of memory management, the author has developed an interactive Java-based tool that uses graphical animation to convey the concepts of various page 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. In this paper, the impact of the tool on student learning is measured and discussed in detail.

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

1 Introduction

Most modern 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. Various page replacement algorithms are discussed in most operating systems textbooks [1][2][3] and the goodness or the performance of page replacement algorithms is measured by the number of page faults generated for a given 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 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 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 critical question is how much this simulator aids student mastery of the content. As such, the opportunity to measure its impact on student learning in the operating systems course at the author’s institute was taken in the fall of 2009.
The results of statistical analysis 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 gives a brief overview of the simulator, section 3 evaluates the simulator’s impact on student learning, section 4 discusses related work, and section 5 draws some conclusions.

2 Overview of the Simulator

In this section, the implemented simulator is briefly described, in order to provide sufficient background for the paper. The detailed description of the simulator can be found in [4].

The simulator supports various types of page replacement algorithms including Optimal, First-In-First-Out (FIFO), Least Recently Used (LRU), Most Recently Used (MRU), Clock, Enhanced Clock (or Third Chance), Least Frequently Used (LFU), and Most Frequently Used (MFU) replacement algorithms. The simulator is written using Java 6 and its main operating modes used in the experimental evaluation are simulation and practice modes. Each mode is briefly described below.

2.1 Simulation Mode

Fig. 1 shows a snapshot in simulation mode. In this mode, through the control buttons (i.e., “Start”, “Stop”, “Next”, and “Reset” buttons), the user can start and stop the animation whenever he wishes, and he can watch the animation straight through from the beginning until the end, or watch it step-by-step.

In Fig. 1, the state of the simulator at virtual time 5 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. 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 virtual time of a process 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 3 is referenced at virtual time 5. Since page 3 is referenced for the first time and all of the...
frames are occupied, it is necessary to replace one of the current three frame pages, the victim page, with page 3. 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 selected algorithm, which varies from algorithm to algorithm.

Also, depending on the algorithm being simulated, the second column of the table in the middle panel may contain some information associated with the page in each frame. For example, in the case of the FIFO algorithm, the second column is the time each page is loaded and in the case of the clock algorithm, the second column of the table is a reference bit of value 0 or 1 indicating that the reference bit is off or on. In the case of clock and enhanced clock algorithms, a pointer indicating that the page in the entry is the oldest page is also displayed.

Once the victim page is selected and replaced, the simulator will display the cost of the current page fault, which is calculated based on whether the victim page has 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. At this point, the table in the bottom panel will be updated with the data from the current virtual time. Then virtual time will increment to the next virtual time.

2.2 Practice Mode
In practice mode, the user can test his understanding of the concepts of page replacement algorithms in two ways: One-step-at-a-time and All-at-once. Fig. 2 shows a snapshot at virtual time 5 in One-step-at-a-time practice mode; at this point, the simulator is waiting for the user input about virtual time 5. Once the user inputs data for virtual time 5 in the left table and clicks the “Submit” button, the simulator will check whether the answer is correct or not. The graphic in the middle indicates the user input for virtual time 4 was correct; if the user input was incorrect then the graphic would say it was wrong and highlight the incorrect answers in the right table. After indicating if the answer is correct, the data in the left and the bottom tables is updated to reflect the correct data.

Fig. 2: A snapshot of the simulator in One-step-at-a-time practice mode
The screen shot of the simulator in All-at-once practice mode is similar to the one in One-step-at-a-time practice mode except that the left and the right tables in the middle area will be disabled whereas the table at the bottom will become editable. Also, the “Submit” button at the bottom will be enabled instead of the one under the left table, and the graphic in the middle area will not be displayed. In All-at-once practice mode, 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.

3 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. 3 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. 3.

The results are shown in Fig. 4 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.

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 α = 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 α = 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 α = 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 engaged 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. For example, Byrne et al. 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 traditional text". 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.

4 Related Work

Visualizations and animations have been used in many areas of computer science education. A number of visualization and animation tools have been created, especially in the area of algorithms and data structures. 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 and the ability to control the pace of the visualization 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 as part of their lecture in an operating system 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 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 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 reference bit is inspected at the hand’s location. If the reference bit is 0, the new page is put in place of the page the clock hand points to; otherwise, the reference 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, 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 mode 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
The author has developed an interactive Java-based simulator that demonstrates the concepts of various page replacement algorithms through animation. The simulator can run in simulation and practice modes. 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.

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 both simulation and 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: