A Process-Scheduling Simulator Based on Virtual Reality Technology

Marcelo de Paiva Guimarães
Open University of Brazil-UNIFESP / Faccamps Master Program
Federal University of São Paulo / Faccamp
São Paulo, São Paulo, Brazil
Email: marcelodepaiva@gmail.com

Vagner Scamati
Faccamps Master Program
Campo Limpo Paulista, São Paulo, Brazil
Email: vagnerscamati@hotmail.com

Mário Popolin Neto
Federal Institute of São Paulo
Araraquara, São Paulo, Brazil
Email: mariopopolin@ifsp.edu.br

Valéria Farinazzo Martins
Faculty of Computer and Informatics
Mackenzie Presbyterian University
São Paulo, São Paulo, Brazil
Email: valfarinazzo@gmail.com

Diego Roberto Colombo Dias
Computer Science Department
Federal University of São João Del Rei
São João Del Rei, Minas Gerais, Brazil
Email: diegodias@ufsj.edu.br

José Remo Ferreira Brega
Computer Science Department
So Paulo State University
Bauru, São Paulo, Brazil
Email: remo@fc.unesp.br

***Abstract*—A process-scheduling algorithm is a fundamental operating system function that manages the assignment of CPU (Central Processing Unit) processes. It aims to make the system efficient, fast, and fair, allowing as many processes as possible to make the best use of the CPU at any given time. Understanding scheduling algorithms and their impact in practice is a challenging and time-consuming task for students, given the differing features of each operating system. Operating system simulators are usually based on either a command line or bi-dimensional front-end interface. This paper presents a process-scheduling simulator based on virtual reality technology that allows students to develop their empirical skills while learning process-scheduling algorithms. The simulator allows the learner to navigate inside a computer motherboard and interact with its components in multi-projection environments such as CAVE-like systems (Cave Automatic Virtual Environment), on a desktop, in a Web browser, or using an HMD (head-mounted display) such as Google Cardboard.**

Keywords: Scheduling Algorithm; Simulator; Operating System; Virtual Reality

I. Introduction

Operating systems have undergone several changes in recent years to suit the demands of new equipment from mobile devices to supercomputers. Nevertheless, some general have been sustained, such as the process scheduler, which decides which process (commonly defined as an instance of a program in execution) will use the CPU (Central Processing Unit) [1], [2], [3]. The algorithm used to make this decision is called the scheduling algorithm. Understanding how process-scheduling algorithms work is fundamental for any computer science student.

Most modern computers have multiprogramming capability (many processes can share the computer's resources in a controlled and safe way). Their ability to operate successfully depends heavily on process-scheduling algorithms to maximize CPU utilization. Switching the CPU between processes is computationally expensive, as it takes time to execute the administration. Reallocating the CPU from one process to another requires performing a context switch, which involves saving and loading registers and memory maps, and updating various tables and lists. Context switches occur in supervisor mode, which the kernel runs and which provides access to memory locations and all other system resources. A process-scheduling algorithm must be sensitive to this computational administration cost.

The process-scheduling algorithm is fundamental to the success of an operating system. Learning these algorithms is a challenging task for computer science students, who must have to learn how the hardware works as well as a given system's features in order to master this content. It requires persistence on the part of students and teachers, appropriate teaching strategies and study methodologies, a capacity for abstract thinking, and motivation.

This paper presents an immersive, interactive 3-D process-scheduling simulator for a single CPU that runs on desktops and head mounted displays (HMD) such as Google Cardboard [4] or in multi-projection environments such as CAVE-like systems (Cave Automatic Virtual Environment) [5], [6]. Students can experiment with various process-scheduling algorithms in this virtual reality environment.

Until recently, cost has been a barrier to the effective adoption of virtual reality simulators in an educational context, and moreover developers had to make a new version of the content for each device. However, technological advances have made it possible to create virtual reality laboratories that are affordable for educational institutions [7]. Equipment such as Google Cardboard and appropriate software can now be purchased in the marketplace. Free and open-source software of very high quality is also now available.

A key goal of this virtual reality simulator is to give students the flexibility to use it for self-study (at home or in class) or group study (in class). Students can run the simulation on

978-1-5090-4320-0/16/$31.00 ©2016 IEEE

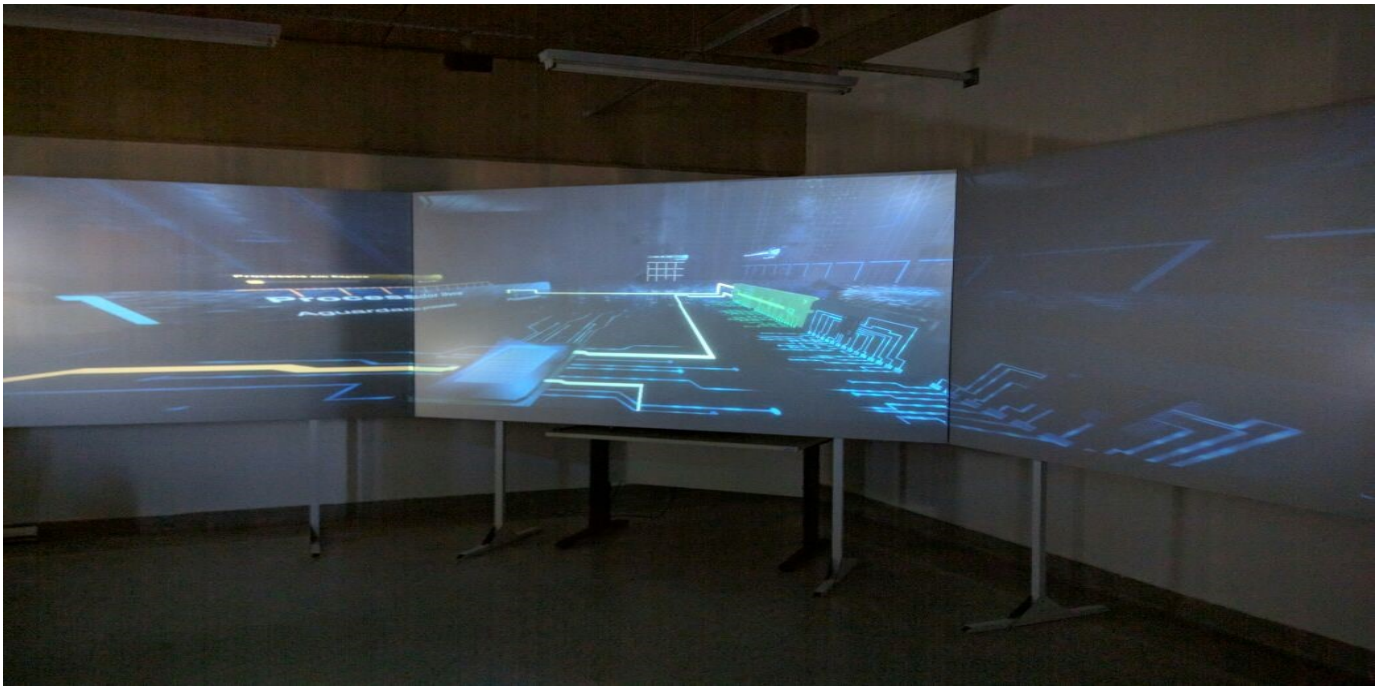

Fig. 1. The virtual reality scheduling simulator is run in a mini-CAVE.

standalone devices, pursuing course objectives at their own pace, or work with other students and their teacher in a multi-projection environment. Using this simulator, students "enter" a motherboard and visualize, manipulate, and exploit the scheduling process in real time. This simulation activity can strengthen, facilitate, and consolidate students' understanding of scheduling concepts. Importantly, such activity must be supported by a well-defined pedagogical strategy to ensure students understand the content.

The simulator has great potential to improve the teaching of process-scheduling algorithms, turning learning into a situation of exploration, discovery, observation, and construction of knowledge, while creating for students a new form in which to understand the content. This approach directly affects students' cognitive processes, aiding their comprehension by providing practical experience of abstract concepts.

Figure 1 shows the virtual reality process-scheduling simulator being executed. In this example, the application is running in a mini-CAVE [8] environment composed of three side walls with passive stereoscopic 3-D (three-dimensional) function and supporting multiple interaction devices, such as Microsoft Kinect, Wii Remote, and keyboard/mouse. The interaction fulfills one of the basic requirements of virtual reality applications: that the simulation reacts to the user's actions in a suitable manner (i.e., the perspective reflects the position and direction of the user). The environment consists of a six-node computer cluster with 3-D graphics cards and six high-definition projectors.

The remainder of the paper is organized as follows: Section II presents an overview of process-scheduling algorithms. Section III surveys the background work related to this research. Section IV explains the process-scheduling simulator's architecture and provides details of its implementation. Section V presents conclusions and plans for future works.

II. Process scheduling algorithms

Processes can be running, waiting, or ready to run (classical states). When one or more processes are ready to run, the operating system must decide which of them will be executed first. Scheduling algorithms for the earliest operating systems, based on punched cards and tape drives, were simple: The system would just run the next job on the tape or card reader. In multiuser and time-sharing systems, often combined with batch jobs in the background, the scheduling algorithm is more complex [3]. Each process-scheduling algorithm is tailored to an operating system. Some criteria used for comparing these algorithms are:

- Fairness: CPU usage is equally distributed among the processes;
- Efficiency: CPU must be kept as busy as possible and no CPU cycles should be wasted;
- Response time: amount of time it takes to start responding from time of submission;
- Turnaround: amount of time it takes to complete a process; and
- Throughput: rate at which processes are completed per unit of time.

In general, all scheduling algorithms aim to meet these criteria and others. Although all of them are desired by all systems, some of them are more important for some operating systems than for others. For example, real-time systems require predictable behavior in terms of guaranteed deadlines [9].

One complication that the design of a schedulers should take into consideration is that each process is unique and unpredictable. Some spend most of their time waiting for input/output (I/O) files, while others would use the CPU for hours if given the chance. When a process is consuming CPU cycles is called a CPU-burst I/O is called an I/O-burst. When the scheduler launches the execution of a process, it never knows for sure how long it will take. A process cannot run too long; almost every computer has a clock mechanism (clock) that causes an interruption periodically. With each clock interrupt, the operating system takes control and decides whether the process can continue running or has already been given enough CPU time. In the latter case, the process is suspended and the CPU is given to another process.

In a modern computer, processes can be in one of several states, defined as the current activity of the process. A process is in the running state when it is currently being executed. One process is allocated to the CPU at a time. A process is in the ready state if it is waiting to use the CPU. It is possible to have more than one process at the same moment (waiting queue). A process is in the waiting state when it is expecting an event to occur, such an I/O operation. It is possible to have more than one active process (waiting queue). Process-scheduling algorithms can be classified according to how they deal with clock interrupts: Scheduling is non-preemptive when a process in the execution state cannot be taken away from the CPU (the process keeps running until it voluntarily blocks or until it finishes), and is preemptive when the process can be CPU.

Operating systems may feature up to three distinct scheduler types [10], [3], [11].

- Long-term scheduler (also known as a job scheduler): Decides which programs are admitted to the ready queue. This scheduler selects processes from a secondary storage device like an SSD (solid-state drive) and loads them into memory for execution. This scheduler aims mainly to achieve a balance between I/O-bound processes (processes that spend more time doing I/O operations than computations) and CPU-bound processes (processes that spend more time doing computations than I/O operations);
- Short-term scheduler (also known as a CPU scheduler). Assigns the CPU to a process from the ready queue. This type makes scheduling decisions much more frequently than long-term or medium-term schedulers. Can be preemptive or non-preemptive; and
- Medium-term scheduler: it is part of the swapping mechanism which removes processes from the memory and places them on secondary memory (such as a SSD) or vice versa, reducing the degree of multiprogramming..

III. Related Work

The use of simulators to teach scheduling algorithms is not new; they are commonly based on command line programs or a 2-D (bi-dimensional) front-end. SOsim [12], [13] is an example of a simulator that illustrates the concepts and techniques found in modern operating systems, such as multiprogramming, processes, scheduling, and memory management. This tool allows students to visualize CPU and main-memory sharing in a dynamic and animated way via a 2-D interface.

Imai and Takeichi (2015) describe a flash-based simulator that supports the algorithms known as First Come-First Served, Shortest Processing Time First, and Round Robin. This is a 2-D simulator designed to run on Web browsers. The learner chooses the algorithm, and the system simulates the execution. It presents the departing time(s) and processing time(s) for each process and the average value of the total response times. The authors perform a quantitative/qualitative evaluation of its practical utilization.

Diaz (2007) presents a scheduling simulator suited to simulating real-time scheduling algorithms. It uses a 2-D front-end to determine the parameters of the processes and to view simulation results. Another 2-D process-scheduling simulator was designed by Robbins and Robbins (1999) that allows students to explore various algorithms and their settings. Based on this, they created a Java applet. Sukanya (2007) presents a scheduling simulator that uses 3-D graphic animation to illustrate concepts related to the algorithms for a single CPU. Supporting various scheduling algorithms, it has a simulation mode that allows students to observe algorithm execution step by step and a practice mode that lets students make their own scheduling decisions.

The simulator presented in this paper is different from these others in several respects, including the degree of immersion supported (desktop, browser, HMD, or multi-projection); the fact that learners can navigate in 3-D space or on a 2-D front-end; offering the option of performing activities in groups or individually; and interaction with scheduling processes by means of input devices such as Kinect, Wii Remote, keyboard, and mouse. No special knowledge is required for teachers and students to manipulate the interface. These features make this tool more than a simple viewer of 3-D objects; it allows the teacher to plan and execute lessons according to his or her educational goals. Teacher and students can navigate freely inside a motherboard, with the ability to choose what they want to view and interact with. The user interaction promotes changes in the environment, generating feedback. For example, the student can choose to view a Process Control Block (PCB), which stores such information about a process as CPU registers, memory management information, program counters, and accounting information. PCBs differ from process to process and contain all the necessary information for representing a process. It is a Short Term Schedule.

Until a few years ago, virtual reality simulations in an educational context required expensive computers to handle the computational requirements, and the amount of available software was limited. Now, these applications can be used by educational institutions in laboratories at an affordable cost or even run on personal computers. Moreover, virtual reality simulations do not constrain users to simulations within the ”real world”; rather, they amplify the user’s physical and temporal perceptions. Such simulations enable users, for example, to navigate ”within” a computer.

IV. The Virtual Reality process scheduling simulator

The purpose of the virtual reality process-scheduling simulator is to help students learn about CPU allocation and improve their understanding of algorithms' parameters. It was designed to develop students' empirical skills while they are learning scheduling algorithms. For example, the teacher can give students a hypothesis involving process scheduling and ask them to devise experiments to support or disprove the hypothesis. Figure 2 depicts the architecture of this tool:

- Virtual Reality Process-Scheduling Simulator: a virtual reality application that simulates process-scheduling algorithms;
- Interaction: allows teacher and students to manipulate the simulation using input devices such as Kinect, Wii Remote, keyboard, and mouse. Interaction aims to promote engagement and immersion in the simulation;
- Navigation: allows teacher and students to move through the motherboard components (i.e., memory, CPU, and I/O device); and
- Visualization: allows teacher and students to envisage the motherboard components using desktop, HMD, or a multi-projection device.

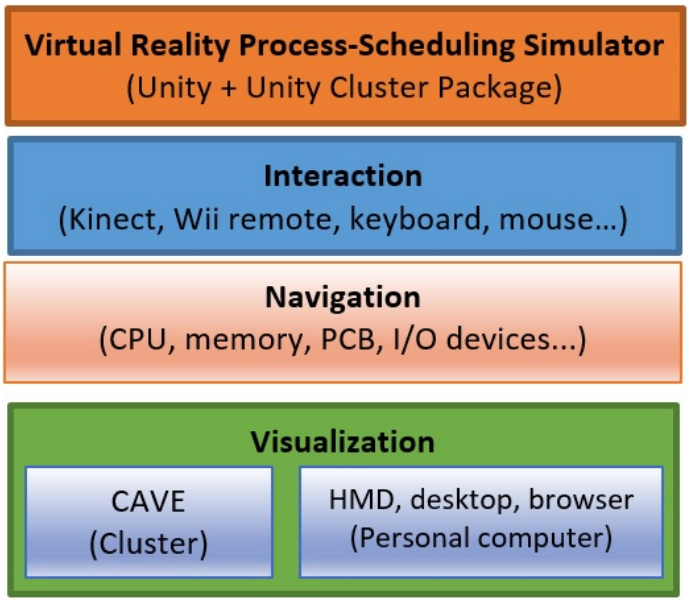

Fig. 2. Simulator: architecture overview

Recent advances in game engines have been accompanied by the creation of sophisticated virtual reality applications. It is now possible to run fully immersive and interactive virtual reality applications on personal computers with commodity graphics cards. The main advantages of using this virtual reality simulator for educational purposes are:

- Higher motivation of teachers and students in general;
- Students can recreate experiments themselves at home;
- Macroscopic and microscopic visualization (a computer can be observed as a CPU);
- Students are encouraged to become active during lectures, since it requires interaction;
- Student creativity and experimentation are encouraged;
- Students can learn at their own pace;
- Students can visualize situations and concepts which could not otherwise be seen;
- Parameters that cannot be changed in the real world, such as timescale, can be altered during a simulation; and
- More efficient illustration of features and processes compared to other multimedia formats.

The success of the simulator presented depends on the cycle depicted in Figure 3. Initially, the scheduling algorithm is chosen by the teacher or a student. Afterward, the user creates the processes. This includes defining their parameters. Finally, the simulation starts. Each process runs for a randomly generated amount of time. From this moment, the user is free to navigate and visualize the motherboard, and interact with the processes.

During navigation, the user can visualize messages and/or listen to descriptions of the current activity and of process transitions of the algorithm (e.g., "A new process is going to the CPU," or "The PCB of the current process in execution is saved." Graphic animations [18] are used to illustrate step by step the processes' state transactions and the movement of processes from one component to another (e.g., from the CPU to the hard drive through the bus system). It is possible, for example, to watch a process moving from the CPU to an I/O device and changing state from running to waiting. It becomes easier to understand what is going on inside the scheduling algorithm and why, at any given time, a particular process is going to run the CPU, or not, as the case may be. Also, a guide is available during the simulation to provide information about the controls and scheduling algorithms. The simulator implements the usual characteristics of an interactive navigation application, such as collision detection.

When the user is creating the processes, he or she can determine the parameters of each according to the algorithm chosen (Figure 4). Examples of these parameters would include priority in the case of a priority-scheduling algorithm or execution time for a Shortest-Job-First (SJR) scheduling algorithm. In all cases, it is possible to determine whether a process is CPU-bound or I/O-bound. The teacher can configure the algorithm and process parameters to simulate particular situations. If a process has low priority (priority-scheduling algorithm) within a set of processes, it will suffer starvation (be denied CPU time). A possible solution to starvation, in this case, is to choose a scheduling algorithm with a priority queue that increases the priority of processes that have been waiting in the ready queue for a long time.

The simulator is designed to be able to add new scheduling algorithms easily, requiring only a new routine to reorder the ready queue according to the criteria. The algorithms supported are:

- First Come-First Served (FCFS): processes are scheduled in the order in which they arrive. It can result in very

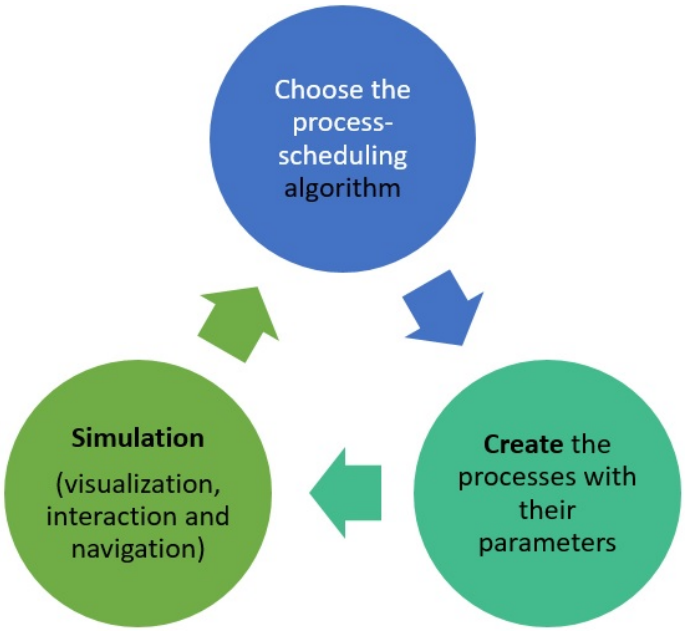

Fig. 3. Simulator cycle.

Fig. 4. A user creates processes and determines their parameters.

long average wait times, particularly if the first process allocated to the CPU takes a long time;

- Round Robin: similar to FCFS. However, each process is assigned a limited amount of CPU time, called a time slice or quantum. If too short, a quantum causes too many context switches; if too long, it can cause poor response time; and
- Priority scheduling: a process is allocated CPU time according to its assigned priority.

The simulator uses the classic three-state model (ready, running or waiting) and it is possible to visualize all process transitions (ready-to-run, run-to-ready, run-to-wait and wait-to-ready), including the moment when the PCB is saved. Figure 5 depicts a user visualizing the contents of a PCB.

The simulator was developed using the Unity game engine, which is used to create games for desktop platforms, mobile devices, Web applications and consoles. Its run-time engine is integrated with a set of tools that facilitate rapid workflows and

Fig. 5. A user visualizes the contents of a PCB.

enable interactive 3-D content. One of the most popular game engines, Unity features a custom rendering engine, extensive documentation and an easy-to-use editor [19]. The simulation version that runs on the mini-CAVE (multi-projection) was implemented using the Unity Cluster Package[20]. This package provides reusable drag-and-drop components, which were used to develop the scheduling simulator for PC clusters, allowing exploitation of Unity's multi-projection system development capabilities (Figure 1). The PC cluster gives multiple views of the same visual data set. Each node has access to the entire data set, then independently determines how much of the data set is visible given its assigned viewing frustum, and renders only that part. The Unit Cluster Package component allows the simulator to run a multi-projection system with passive support for stereoscopy, perspective correction according to the user's viewpoint and access to specific servers to provide device-independent features. There is also a version for devices based on Google Cardboard, and an HMD solution provided through the use of smartphones for image generation. The interaction is captured using a Bluetooth control integrated into the Unity3D environment. Figure 6 depicts the Google Cardboard execution. The same content can be visualized on this device, albeit with limited interaction resources, immersion, and involvement.

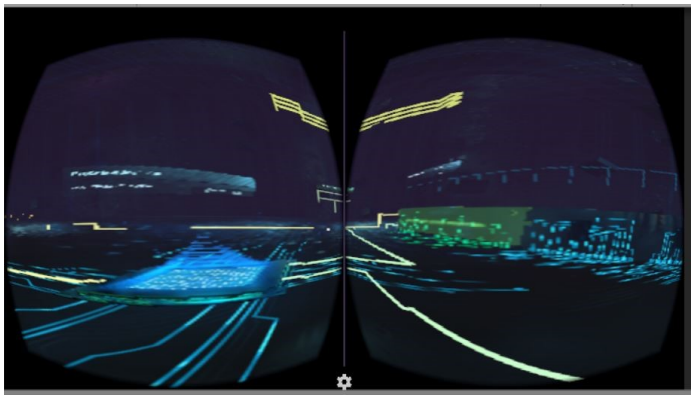

Fig. 6. Google Cardboard execution (left and right eyes).

V. Conclusions

There is growing interest in using simulators to facilitate the teaching/learning process in operating system courses. However, an efficient simulator that is affordable and easy to use requires a solution that allows teachers to focus on educational content and not on technological aspects of classroom materials.

This paper presents a virtual reality application that simulates process-scheduling algorithms. It can be viewed in many different media: in the immersive and interactive environment of a CAVE, on a desktop, in a Web browser, or using an HMD. This application can help students understand scheduling algorithms and get familiar with the basic concepts that underpin them. Through animation and interaction, the user can "enter" the motherboard and perceive details that would not be available to them in the real world. For now, only certain algorithms are available, but the simulator has the flexibility to incorporate additional scheduling algorithms and other features such as multiprocessing management.

In future, we plan to improve the simulator by adding new algorithms. We also plan to add new content, such as memory management, and conduct experiments to determine the simulator's effect on student learning.

Acknowledgment

The authors gratefully acknowledge support from FAPEMIG, Capes and CNPq.

References

[1] W. Stallings, *Operating Systems: Internals and Design Principles*. Pearson, 2014.

[2] P. Deitel and C. D. R, *Operating Systems*. Pearson, 2003.

[3] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[4] Google, "Google cardboard – google vr," https://vr.google.com/cardboard/index.html, 2016, Ĺast acess: June 2016.

[5] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The cave: Audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, no. 6, pp. 64–72, Jun. 1992. [Online]. Available: http://doi.acm.org/10.1145/129888.129892

[6] S. Manjrekar, S. Sandilya, D. Bhosale, S. Kanchi, A. Pitkar, and M. Gondhalekar, "Cave: An emerging immersive technology – a review," in *Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on*, March 2014, pp. 131–136.

[7] L. Carozza, F. Bosché, and M. Abdel-Wahab, "Robust 6-dof immersive navigation using commodity hardware," in *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '14. New York, NY, USA: ACM, 2014, pp. 19–22. [Online]. Available: http://doi.acm.org/10.1145/2671015.2671115

[8] D. C. Dias, A. La Marca, A. Moia Vieira, M. Neto, J. Brega, M. de Paiva Guimaraes, and J. Lauris, "Dental arches multi-projection system with semantic descriptions," in *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on Virtual Systems and Multimedia*, ser. VSMM 2010, no. ISBN 978-1-4244-9027-1. Seoul: IEEE Xplore, oct. 2010, pp. 314 –317.

[9] L. Sha, T. Abdelzaher, K.-E. én, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, no. 2-3, pp. 101–155, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:TIME.0000045315.61234.1e

[10] I. Dhotre, *Operating Systems*. Technical Publications, 2008. [Online]. Available: https://books.google.com.br/books?id=8eh2ROK2BK4C

[11] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 8th ed. Wiley Publishing, 2008.

[12] L. P. Maia, F. B. Machado, and A. C. Pacheco, Jr., "A constructivist framework for operating systems education: A pedagogic proposal using the sosim," in *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '05. New York, NY, USA: ACM, 2005, pp. 218–222. [Online]. Available: http://doi.acm.org/10.1145/1067445.1067505

[13] L. P. Maia and A. C. Pacheco, "A simulator supporting lectures on operating systems," in *Frontiers in Education, 2003. FIE 2003 33rd Annual*, vol. 2, Nov 2003, pp. F2C–13–17 Vol.2.

[14] Y. Imai and K. Takeichi, "Development and evaluation of adobe flash based cpu scheduling simulator executable on major multiple web browsers," in *Intelligent Networking and Collaborative Systems (INCOS), 2015 International Conference on*, Sept 2015, pp. 149–155.

[15] A. Diaz, R. Batista, and O. Castro, "Realtss: a real-time scheduling simulator," in *Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on*, Sept 2007, pp. 165–168.

[16] S. Robbins and K. A. Robbins, "Empirical exploration in undergraduate operating systems," in *Proc. 30th SIGCSE Technical Symposium on Computer Science Education*, 1999, pp. 311–315.

[17] S. Suranauwarat, "A cpu scheduling algorithm simulator," in *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, Oct 2007, pp. F2H–19–F2H–24.

[18] J. Stasko, A. Badre, and C. Lewis, "Do algorithm animations assist learning?: An empirical study and analysis," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, ser. CHI '93. New York, NY, USA: ACM, 1993, pp. 61–66. [Online]. Available: http://doi.acm.org/10.1145/169059.169078

[19] R. Beimler, G. Bruder, and F. Steinicke, "Smurvebox: A smart multi-user real-time virtual environment for generating character animations," in *Proceedings of the Virtual Reality International Conference: Laval Virtual*, ser. VRIC '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:7. [Online]. Available: http://doi.acm.org/10.1145/2466816.2466818

[20] M. P. Neto, D. R. C. Dias, L. C. Trevelin, M. de Paiva Guimarães, and J. R. F. Brega, *Computational Science and Its Applications – ICCSA 2015: 15th International Conference, Banff, AB, Canada, June 22-25, 2015, Proceedings, Part V*. Cham: Springer International Publishing, 2015, ch. Unity Cluster Package – Dragging and Dropping Components for Multi-projection Virtual Reality Applications Based on PC Clusters, pp. 261–272. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21413-9_19