A Virtual Reality Simulator to Assist in Memory Management Lectures

Luiz Felipe Santos Freitas[1], Alex Sandro Rodrigues Ancioto[1], Rita de Fátima Rodrigues Guimarães[1,2], Valéria Farinazzo Martins[3], Diego Roberto Colombo Dias[4], and Marcelo de Paiva Guimarães[1,5(✉)]

[1] Computer Science Master Program, Unifaccamp, Campo Limpo Paulista, Brazil
luiz.freitas@ifms.edu.br, alexancioto@gmail.com, rita.guimaraes@gmail.com, marcelodepaiva@gmail.com
[2] Applied Linguistics (Institute of Language Studies/Unicamp), Campinas, Brazil
[3] Faculty of Computing and Informatics, Mackenzie Presbyterian University, São Paulo, Brazil
valfarinazzo@hotmail.com
[4] Computer Science Department (UFSJ), São João del Rei, Brazil
diegocolombo.dias@gmail.com
[5] Núcleo de Ensino a Distância (UAB/Reitoria), UNIFESP, São Paulo, Brazil

Abstract. Virtual reality technology can assist the teaching-learning process via concrete concepts that follow a sequence of steps to accomplish a task in a three-dimensional space, as proven in the literature (e.g., operating a vehicle or disaster relief simulation). However, it is not clear whether virtual reality can also enhance situations involving concepts that are not inherently related to a three-dimensional space, such as the execution of a computer algorithm. This paper presents an immersive and interactive virtual reality simulator that can aid in the teaching of the memory management functionality of operating systems, including single, fixed and dynamic contiguous techniques, and the non-contiguous technique of paging. Learners are immersed inside a computer motherboard to learn memory management functionality of operating systems. They use a head-mounted display such as 3D virtual reality headsets and can interact with the environment using eye-gaze tracking and a joystick. We also present a case study in which 80 students were divided into two groups to evaluate the simulator. Our results indicate that using our simulator is a more effective approach for teaching memory management concepts than expositive classes. The functionality and usability tests results highlighted the positive aspects of the simulator and improvements that could be made.

Keywords: Computer science · Learning environment · Virtual reality · Memory management · Operating system

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12255, pp. 810–825, 2020.
https://doi.org/10.1007/978-3-030-58820-5_58

1 Introduction

Virtual reality has the potential to immerse a learner into situations that would otherwise be difficult (e.g., driving a car [1], a flight simulator [2], arthroscopic training [3], sports training, and education [4]) or would not be feasible (e.g., visiting the interior of a volcano in eruption [5] or navigating within a macromolecular system [6]). It can modify traditional teaching-learning processes, resulting in practical, interactive and fun learning experiences. The scientific literature shows that virtual reality can enhance the learning experience via concrete situations in which learners follow a sequence of steps to accomplish a task (i.e., actions that can be experienced through the senses). Examples include designing a house [7], rehabilitation training [8], welding training [9], operating a vehicle remotely [10] and manipulation of three-dimensional (3D) geometric objects [11, 12]. These examples have in common the fact that they require maneuvers in 3D space, and are experiences involving problems and context that are identical to those in the real world [13]. However, there is also teaching-learning content that is not related to a 3D space, such as the execution of algorithms (i.e., things that do not have a physical existence).

The ability to grasp and manipulate abstract ideas is a vital element of computer science [14, 15]. For example, students need to master abstract concepts that are intrinsic to algorithm design. We study the use of virtual reality in the teaching of memory management in operating system courses; this involves abstract content, and is mandatory in many computer science, information systems and computer engineering curricula. Memory management is one of the most essential functions of an operating system, and is responsible for allocating and managing the computer's main memory [16–18]. All software instructions, data and control instructions are stored in some form of memory. The memory management function monitors the status of each memory position, i.e. whether it is allocated or free, with the aim of achieving effective and efficient use of the main memory. The quantity of memory and the allocation technique used vary according with the type of computer (e.g., supercomputer, desktop, mobile, smart card).

The main memory techniques can be contiguous, meaning that the running process is loaded entirely in memory, or non-contiguous, meaning that the process can be partially loaded [16–18]. Traditionally, a memory management lecturer will follow a textbook, prepare and exhibit slides, and present some theoretical (teacher-centered) exercises [13]. Practical exercises are assigned as homework or projects, and are executed using a real operating system or a simulator based on command-line programs or a two-dimensional (2D) front-end.

This paper presents an immersive and interactive 3D memory management simulator for both contiguous and non-contiguous memory allocation techniques for operating systems. The simulator runs on commodity devices such as desktops and head-mounted displays (HMD) (e.g., Google Cardboard, 3D VR Headsets). Learners can study allocation techniques in situations involving exploration, discovery, observation and the construction of knowledge. The allocation algorithms are visualized and explored using metaphors in the form of 3D elements (e.g., colored boxes). These elements can also represent items within the computer (e.g., processor and controller devices). The simulator can visualize single, static and dynamic contiguous techniques, and non-contiguous paging techniques [16–18].

A key goal of this virtual reality simulator is to promote a (student-centered) discovery approach that gives students the opportunity to enhance their learning and understanding of memory management content via self-study, by means of scenario simulations involving immersion, interactivity and involvement. The simulator offers practical experience of concepts related to memory management. This paper also presents a case study conducted with 80 students, who were randomly assigned into two groups to perform a learning evaluation. Forty of these students formed an experimental group, and were engaged in an introductory memory management lesson using the simulator, while the remainder of the students formed a control group and received instruction in an expositive class. Following the introductory memory management lesson, a post-test was conducted, and a statistical test was performed on the scores of both groups. We hypothesize that memory management concepts can be better assimilated by undergraduate students when the learning process is supported by virtual reality. The students who used the simulator also underwent a functionality and usability test.

The contribution of this paper is three-fold:

- We present an immersive and interactive memory management environment that can assist in memory management classes;
- A new teaching approach based on a virtual reality simulator is validated, rather than simply using the traditional approach to teaching memory management;
- We investigate the use of virtual reality to teach concepts that are not inherently associated with a 3D space.

The remainder of the paper is organized as follows: Sect. 2 discusses prior work related to this research; Sect. 3 describes the methodology used to create and evaluate the simulator; Sect. 4 presents the simulator; Sect. 5 reports on a case study involving our simulator; And Sect. 6 presents the conclusion and suggestions for future work.

2 Related Work

Several efforts have been made towards facilitating the processes of teaching and learning about operating systems. The typical educational approach to assisting these classes is to use project tools such as MINIX [19], GeekOS [20] and ICS OS [21]. This approach allows students to modify the operating system code directly, requiring them to master the programs written in C, C++ and in many cases assembly language.

An alternative approach is to adopt simulators that are based on command line programs or a 2D interface. SOsim [22] is one example that illustrates concepts such as multiprogramming, scheduling, processes, and virtual memory management. This tool uses a 2D matrix to represent processes stored in memory, introducing concepts and techniques using a dynamic and animated 2D interface. Ontko, Reeder and Tanenbaum presented a set of four simulators (scheduling, deadlocking, memory management and file system) called Modern Operating System Simulators (MOSS) [23]. These are based on a command line interface with some graphic elements, and their execution and use are not trivial.

Lopes et al. proposed the Memory Simulator for Teaching of Operating Systems (SIME) [24], which involves a 2D graphical representation of memory management. The student chooses a topic such as memory allocation (contiguous, overlay, static partition, dynamic partition, virtual paged memory) or strategy (first fit, best fit, worst fit, swapping), and the simulator illustrates this. RCOS.java [25] is a 2D java application that simulates a multi-tasking operating system running on simulated hardware. This tool demonstrates the general principles of operating systems (e.g., process and disk management) through controlled animation.

Moreno et al. [26] presented a memory management simulator called MNEME, which uses a 2D interface to assist to teach memory hierarchy and includes topics such as multithreading, hyperpaging, direct and reverse mapping, translation lookaside buffering with levels, age paging control, data loading from server and the definition of an eviction policy. However, it does not simulate memory allocation algorithms. Putchal and Bryant [27] proposed a tutoring system called Synchron-ITS to demonstrate concepts associated with process synchronization and shared memory management. This tool offers a 2D visualization of real-world data from a running Linux operating system, using high-level diagram models and source code examples, enabling students to see the connections between abstract models of synchronization. These authors plan in future work to enable visualizations of the synchronization data structures and algorithms in a 3D representation. ESORV [28] is a process-scheduling simulator based on virtual reality technology that allows students to learn algorithms while navigating inside a computer motherboard and interacting with its components. However, it does not cover the topic of memory management.

Although these simulators aim to assist in courses on operating systems, none of them offer an immersive and interactive learning environment to aid in the teaching of memory management. Moreover, only ESORV offers a 3D interface. The solution presented here is different from the abovementioned approaches in that we address the daunting problem of teaching memory management concepts using virtual reality in a fully immersive and interactive environment. As a result, students are able to take advantage of widely used technologies, thereby improving their learning and understanding. We also present and discuss the results obtained from a case study.

3 Methodology

The simulator was developed using a prototyping technique that was divided into four phases [29]. Phases I, II and III were iterative, and in each of these, a new functionality was identified and incorporated into the system. Phase IV was performed only once. Details of each phase are presented below:

- Requirements (phase I): This task aimed to elicit the requirements for the simulator. We selected 10 teachers of operating systems to fill out a questionnaire which covered the following topics: teacher profile (e.g., age, sex, education); operational systems classes (e.g., topics covered, theoretical/practical classes, educational tools adopted); virtual reality technology (e.g., knowledge, devices, experience in class); and memory management subject (e.g., topics covered, students difficulties).

The results confirmed the existence of a research gap in terms of developing a simulator. We then elicited the relevant requirements.

- Design (phase II): This involved the following basic elements [30]: mechanics, which defined the logic involved (e.g., rules, interaction with the simulator); aesthetics, which were related to the look and feel of the simulator (e.g., metaphors to represent allocation techniques, 3D models, audio, textures, colors); a story that determined the navigation within the environment (e.g., linear or nonlinear navigation, memory management content); and the technology used to create the simulator (e.g., game engine or 3D modeling tool).
- Implementation (phase III): In this step, the simulator was coded and other elements were also created (e.g., 3D models, textures and sounds.
- Evaluation (phase IV): Eighty students were randomly assigned into two groups. The 40 students in the experimental group engaged in an introductory memory management lesson using the virtual reality simulator, while the students in the control group received instruction via an expositive lecture. After this introductory lesson on memory management, a post-test was conducted, and a statistical test was performed on the scores of both groups. We discuss the data analysis and findings of our study below.

4 The Virtual Reality Memory Management Simulator

The main purpose of the simulator was to assist students in learning about the allocation of main memory and in improving their understanding of the advantages and disadvantages of each allocation technique. It was designed so that new techniques could be added easily, requiring only a new routine to allocate the memory according to the new technique. The simulator was developed based on the Unity game engine [31], which is used to create games for desktop platforms, mobile devices, browser-based applications and consoles. Unity projects are exported to diverse platforms natively (e.g. Android, Linux, and Windows). During the simulation, each student can enter into a virtual motherboard and visualize, manipulate and exploit the memory management in real time, pursuing the course objectives at their own pace.

The process of development of the simulator started by using the requirements, drawn from a survey of 10 teachers of operating systems (we were not included). Of these, four teachers (40%) had more than five years of experience of teaching operating systems, five (50%) had between one and five years, and one (10%) had less than one year. All respondents used didactic resources such as transparencies, projectors and acrylic boards. Two (20%) used a simulator, and one (10%) used an open source operating system (Linux). Eight (80%) had just theoretical lectures in their undergraduate course. Seven (70%) reported a lack of educational resources for practical lectures. This lack of innovative material for practical classes motivated us to try and bridge this gap by using an immersive and interactive virtual reality tool to foster learning. The following requirements were identified:

- Environmental requirements (related to the physical environment in which the simulator is deployed):

  - Students should be seated in a safe and secure place in order to avoid possible falls, since once the HMD is deployed, the user loses sight the real physical space, and consequently loses cognitive balance;
  - Students should not make sudden movements with their heads, in order to reduce sickness or dizziness.
- Functional requirements (related to the functionalities of the simulator):
  - The memory is allocated to a process during its creation;
  - An allocation technique is chosen when the simulation starts. If another technique is chosen, the simulation should be restarted. The user is free to choose any allocation technique;
  - Any process can be finalized during the simulation, and the memory allocated must be released. If the operating system process is finalized, all other processes are also finalized;
  - The first process started by the user must be the operating system. Following this, the user can start other processes.
  - A process is allocated only when enough memory is available;
  - Visual and audio messages should be provided to help users during navigation and interaction, for example, an alert insufficient memory to allocate the process;
  - User should be able to visualize the process control block (PCB) to see details of the process;
  - All elements of the simulation should be represented as 3D elements;
  - The simulation must include contiguous and non-contiguous allocation algorithms.
- Non-functional requirements (related to how the simulator should behave and the constraints on its behavior):
  - New techniques should be able to be added in future;
  - It must support fully immersive visualization, interaction and navigation;
  - It must have versions for both desktop and HMDs (multiplatform);
  - It should be easy to use;
  - It should use clear metaphors to represent the abstract concepts in a 3D environment;
  - Each allocation technique must be simulated in isolation.

The simulator was designed to improve the student's empirical skills as they are learning the allocating algorithms. For example, the teacher can formulate a hypothesis about an algorithm and ask the students to run experiments to support or disprove it. Figure 1 depicts the flow chart for operation of the simulator. After initialization, the user chooses a memory allocation technique or the exit option, and the simulation starts. Soon after this, the user starts to navigate and interact with the simulated environment. Users can perform actions such as creating or deleting processes, or visualizing a PCB. They can use a joystick or eye-gaze tracking to interact with the environment. During creation of a process, spaces memory are associated with colors, and audio and visual messages are presented to users according to the allocation technique chosen initially. Users can also interact with the motherboard elements, such as the processor and device controllers, and visual and audio messages describing their operation are presented.

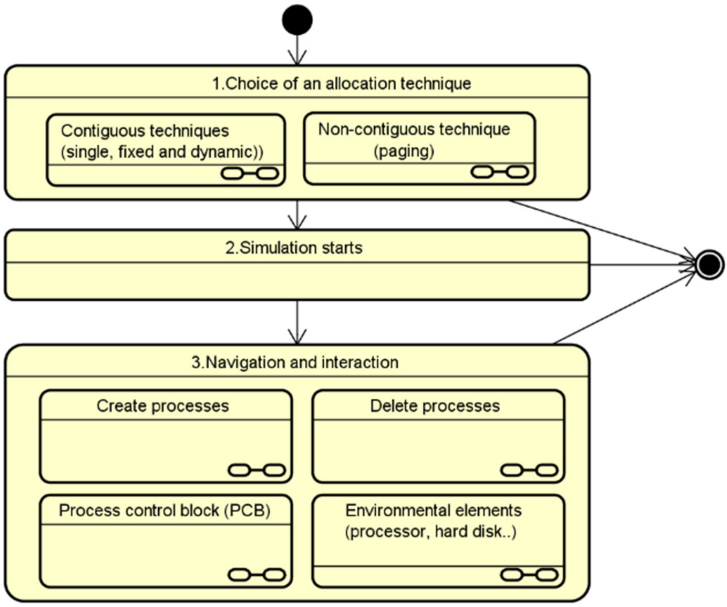


Fig. 1. Flow chart for simulator operation

Figure 2 depicts the simulated motherboard. If the student selects a motherboard element, a realistic computer-generated voice offers an explanation of it, such as "This is the CPU, which performs most of the processing inside the computer". When the help option is chosen, a video clip explains how the simulator works. Figures 2, 3, 4, 5 and 6 have a white point which is the position where the user is looking. This allows the user to interact with an object, triggering an animation for example. This is known as gaze.

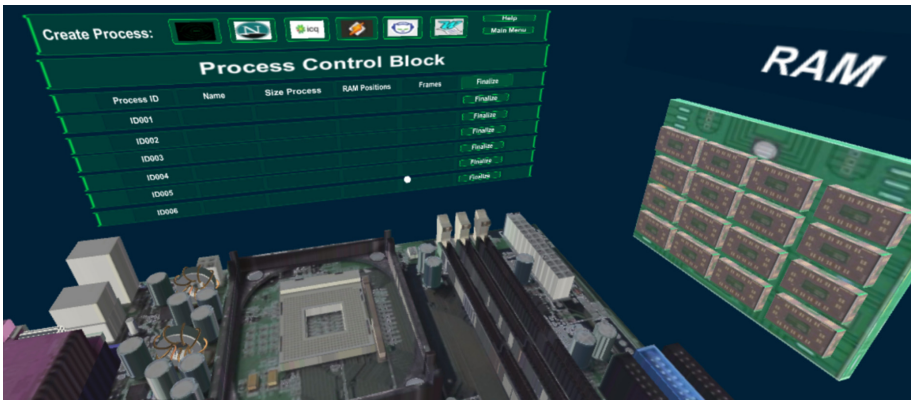
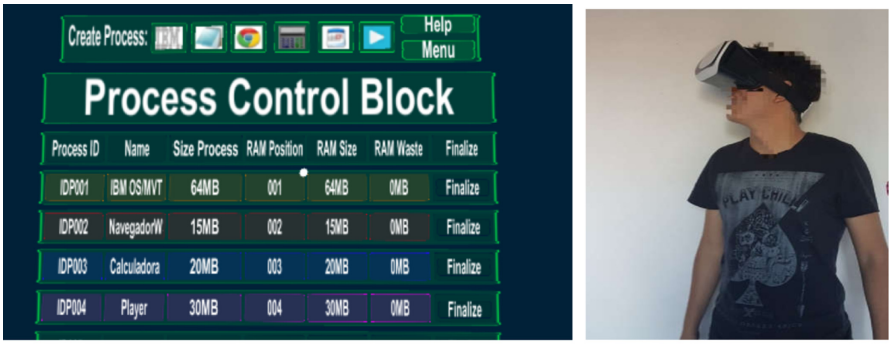


Fig. 2. Memory management simulator

Figure 3(a) depicts a learner visualizing the content of a process control block (PCB), which is a data structure that contains information (e.g., process identification and size) used to manage the process. PCBs differ between processes, and store all the necessary information to represent each one. This data structure stores information about how much memory is used by the schema and what type of memories are used (e.g., page tables, limit registers or segment tables). Figure 3(b) shows a student from Group 2 using a 3D VR Headsets with Bluetooth Remote to run the memory management simulator. The student's face was blurred.



(a) (b)

Fig. 3. (a) A student visualizes the content of a PCB; (b) A student running the 3D simulation.

Figure 4 depicts the execution of the simulator in a 3D VR Headsets, with a different point of view for the left and right eyes.

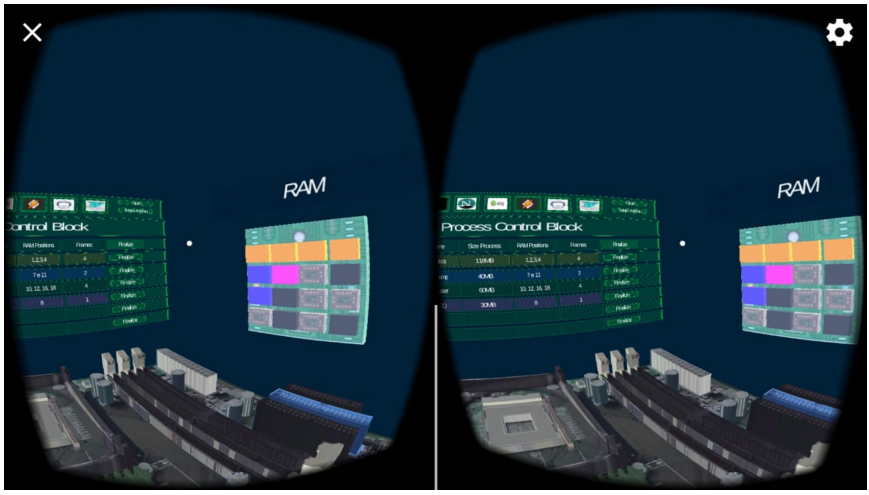


Fig. 4. Simulator running in a 3D VR Headsets (left and right eyes)

In order to illustrate the algorithms, we define metaphors for each situation. This allows us to cover subjects that students normally cannot see or touch but which they need to comprehend in order to understand how memory management works. Woollard [32] argues that these metaphors play an important role in teaching computing and analyzing diverse aspects of their use, and identifies a need for determining the effectiveness and efficiency of particular metaphoric strategies.

The solution adopted here to represent the allocation algorithms was to manipulate dynamic elements based on the student's actions. For example, colored boxes represent the memory area according to the allocation technique chosen. We extended the criteria defined by Mackinlay [33], which were defined to codify a 2D image, in order to create expressiveness, which determines whether the metaphor expresses the desired information, and effectiveness, which determines whether the metaphor explores the human visual system in an effective way. Oberhauser and Lecon [34] proposed an approach that could also be added to this simulator in the future. They created multiple metaphors for visualizing software code structures (e.g., packages and dependencies between classes) in a 3D environment. Their solution has switchable and customizable metaphors, such as visuals (space, terrestrial); groupings (solar systems, glass bubbles or tree-lined cities); connections (pipes, light rays); labels; and background sounds. This is an interesting approach, but it is not common to discuss code details when allocation techniques are being explained (programming would be a prerequisite).

All allocation algorithms implemented here were converted to visual representations using metaphors. Figure 5 depicts the paging allocation technique, in which the address space is broken into blocks of the same size, called pages, and the main memory is divided into small fixed-sized blocks of (physical) memory, called frames. When a process is created, the memory space is represented, and an audio recording explains what is happening. In this case, the user created four processes, and four frames were allocated for the IDP001 process (operating system – shown in orange); four for the IDP002 process (Winamp application - shown in blue); four for the IDP003 process (Napster application – shown in black); and four for the IDP004 process (ICQ application - shown in pink). Memory spaces shown in brown are free spaces. Details about each process can be visualized in the PCB, including the table page.

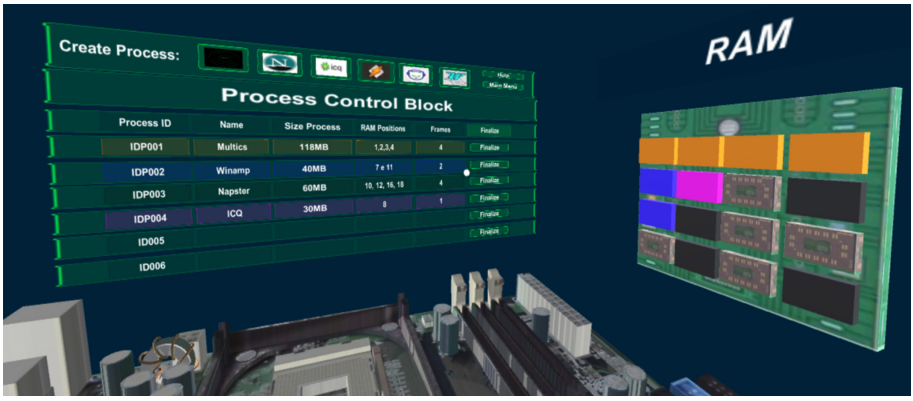


Fig. 5. Paging allocation (non-contiguous technique).

Figure 6 depicts the static partitioned memory allocation technique, in which the memory is divided into a number of separate fixed areas, each of which can hold one process. Each partition may contain unused space (internal fragmentation).

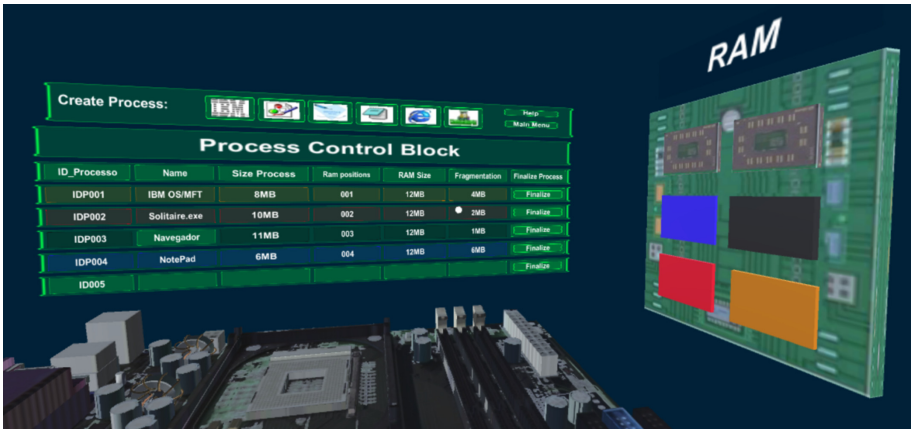


Fig. 6. Static partitioned memory allocation (contiguous technique).

This simulator enables the user to navigate through the motherboard and to observe the memory management technique chosen from various angles with a good spatial view and intuitive interactions, promoting a discovery approach. This solution makes it possible, combining abstract and concrete data in the same virtual reality environment side by side, helping users to understand the links between them.

5 Case Study

An understanding of the students' needs, both in terms of their preferences in the immersive environment and their learning styles, is a key aspect of a successful 3D simulator. In order to validate and improve our simulator, we carried out two randomized experiments: a learning evaluation and a usability evaluation. The study took place in the Federal Institute of Mato Grosso located in the city of Campo Grande, Brazil, with Computer Science degree students. The tests using the simulator were taught in classrooms with one 3D VR Headsets with Bluetooth Remote per student.

5.1 Learning Evaluation

We carried out a learning experiment to evaluate the benefits of using our 3D simulator to teach memory management. We set out to answer the following research question:

Research question: Is our 3D simulator a more effective approach in teaching memory management than an expositive lecture (traditional approach)?

This research question was used as the basis to formulate our hypotheses. Throughout this section, we refer to the approach based on the expositive class as Expositive and that based on the simulator as Simulator.

Hypothesis Formulation: The research question was formalized to allow statistical tests be carried out.

- Null hypothesis (H0): there is no significant difference in efficiency between the two teaching strategies (measured in terms of the student's scores), which can be formalized as follows: H0 = μExpositive = μSimulator.
- Alternative hypothesis, H1: there is a significant difference in efficiency between the two teaching strategies, which can be formalized as follows: H1 = μExpostive ! = μSimulator.

The experiment was broken down into four steps. These steps are listed in chronological order in Table 1.

Table 1. I Stages of learning evaluation

Step	Description	Time
Student profile	All students answered a student profile questionnaire (e.g., age, sex, knowledge of operating systems)	15 min
Division of class	The students were randomly divided into two groups of 40 students.	Free
Learning phase	Group 1: this group went to the laboratory and was taught via an expositive class Group 2: this group went to the laboratory and was taught with the 3D simulator	30 min
Scored test	All students answered twelve questions about memory management	15 min

From the results of the pre-test questionnaire, it was possible to determine the user's profile: all students (100%) had no knowledge of the concepts of operating systems – 38.75% were female; students were between 16 and 53 years old, and half of the users were between 17 and 19 years old.

The class was divided randomly into two groups of 40 (Groups 1 and 2), and we started the lecture (30 min). Group 1 undertook the expositive lecture, and Group 2 used the 3D simulator on an individual basis. Table 2 shows the scores of the students in Groups 1 and 2 in the test (on a scale from zero to 12). The average score for Group 1 was 6.875, and for Group 2 it was 10.425. As shown in Table 2, the 3D simulator lecture method seems to be a more effective approach to introducing students to memory management concepts, and most subjects retained the information imparted to them and performed well on the test.

Table 2. Summary of the scores obtained by Groups 1 and 2 (on a scale of zero to 12)

	Group 1: Expositive class	Group 2: 3D simulator
Median	7	11
Mean	6.875	10.425
Std	3.30	1.48
Max	12	12
Min	2	7

The std for the Group 1 was 3.30, while that for Group 2 was 1.48. The p-value for both was $3.4 * 10^{-8}$ (significance level 0.05). The null hypothesis was rejected since the tests were significant; that is, the change in the student's scores was not random. The results seem to indicate that using the 3D simulator to teach memory management simulator can be an effective approach. Moreover, the teacher who carried out the test reported that the experiment motivated the students, including those who did not usually pay attention during the expositive lecture. The students did not have difficulty interacting with and navigating in the 3D simulator.

5.2 Functionality and Usability Test

After Group 2 had completed the educational test, they also filled out a questionnaire with 12 questions about simulator functionalities and their usability (15 min). The Likert scale [35] was adopted (Strongly disagree, Disagree, Neither agree nor disagree, Agree, Strongly agree). Table 3 gives details of the questionnaire and the results obtained in terms of the user experience, navigation dynamics, and potential execution errors. None of the students left comments (positive, negative or suggestions) about the simulator.

Table 3. System utilization experiences - Group 2

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
1. The simulator has a pleasant and understandable graphical interface	0.0%	7.5%	12.5%	35.0%	45.0%
2. It is easy to understand what to do in the simulator	0.0%	5.0%	5.0%	27.5%	62.5%
3. The simulator interface is very intuitive	0.0%	5.0%	25.0%	20.0%	50.0%
4. It was easy to learn how to use the simulator	0.0%	5.0%	2.5%	27.5%	65.0%
5. The performance of the simulation was good	0.0%	0.0%	0.0%	20.0%	80.0%

(*continued*)

Table 3. (*continued*)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
6. The audio messages worked correctly and were informative	0.0%	0.0%	5.0	17.5%	77.5%
7. When a process is created, it is possible to visualize its memory allocation	0.0%	0.0%	0.0%	17.5%	82.5%
8. It is easy to interact with the elements in the simulator	0.0%	15.0%	2.5%	30.0%	52.5%
9. The video clip that explains how the simulator works is easy to understand and informative	0.0%	2.5%	0.0%	20.0%	77.5%
10. I did not feel discomfort when using the simulator	2.5%	7.5%	2.5%	7.5%	80.0%
11. I felt that I was navigating inside a computer	0.0%	5.0%	0.0%	42.5%	52.5%
12. The simulator helped me to understand memory allocation techniques	0.0%	0.0%	0.0%	17.5%	82.5%
Comments on the simulator (positive or negative aspects, suggestions)					
None of the students left comments					

6 Conclusions

Until recently, the use of virtual reality simulators in an educational context has encountered barriers in terms of the high price of the equipment and a lack of available software. However, technological advances have changed this scenario, allowing teachers to adopt this technology at an affordable cost, for example using mobile devices.

We believe that virtual reality has the potential to provide a myriad of benefits to the learning process, including for subjects that are not inherently related to a 3D space, such as the execution of a memory allocation algorithm. However, to evaluate these benefits, further research is still needed.

This research presents an immersive and interactive simulator for memory allocation. It uses metaphors to represent the execution of the algorithms and the hardware elements. Suitable metaphors were important in increasing the user's perception of directedness, enclosure, engagement, navigation and presence in the motherboard environment. The simulator provided the students with an almost tangible picture of all the concepts and hardware elements involved. Audio media was also used to support this understanding.

An experiment was conducted, and the results showed that the experimental group using the simulator performed significantly better than the control group in terms of achievement in memory management learning. The immersive and interactive simulator environment was able not only to deepen students' memory of the subject, but also to compensate for the shortcomings of a traditional expositive class. We suggest that virtual reality simulators can be used to teach concepts associated with memory management. We also conducted a usability test to evaluate the simulator, and the results highlighted the positive aspects of the simulator and improvements that could be made.

In future, we plan to improve the simulator by adding new allocation techniques and creating better 3D models. We also plan to conduct experiments in multi-projection environments such as systems similar to the CAVE (Cave automatic virtual environment).

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Ihemedu-Steinke, Q.C., Erbach, R., Halady, P., Meixner, G., Weber, M.: Virtual reality driving simulator based on head-mounted displays. In: Meixner, G., Müller, C. (eds.) Automotive User Interfaces. HIS, pp. 401–428. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-49448-7_15
2. Oberhauser, M., Dreyer, D.: A virtual reality flight simulator for human factors engineer-ing. Cogn. Tech. Work **19**(2–3), 263–277 (2017). https://doi.org/10.1007/s10111-017-0421-7
3. Karahan, M., Kerkhoffs, G.M.M.J., Randelli, P., Tuijthof, G.J.M. (eds.): Effective Training of Arthroscopic Skills. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-44943-1
4. Staurset, E.M., Prasolova-Førland, E.: Creating a smart virtual reality simulator for sports training and education. In: Uskov, V.L., Howlett, R.J., Jain, L.C. (eds.) Smart Education and e-Learning 2016. SIST, vol. 59, pp. 423–433. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39690-3_38
5. Boudreaux, H., et al.: V-volcano: addressing students' misconceptions in earth sciences learning through virtual reality simulations. In: Bebis, G., et al. (eds.) ISVC 2009. LNCS, vol. 5875, pp. 1009–1018. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10331-5_94
6. Salvadori, A., Frate, G.D., Pagliai, M., Mancini, G., Barone, V.: Immersive virtual reality in computational chemistry: applications to the analysis of QM and MM data. Int. J. Quantum Chem. **116**(22), 1731–1746 (2016)
7. Wang, X., Schnabel, M.A. (eds.): Mixed Reality in Architecture, Design, and Construction. Springer, Dordrecht (2009). https://doi.org/10.1007/978-1-4020-9088-2
8. Zheng, J., Shi, P., Yu, H.: A virtual reality rehabilitation training system based on upper limb exoskeleton robot. In: 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), vol. 1, pp. 220–223 (2018)
9. Xie, B., Zhou, Q., Yu, L.: A real-time welding training system base on virtual reality. In: 2015 IEEE Virtual Reality (VR), pp. 309–310 (2015)

10. Chin, C.S., Kamsani, N.B., Zhong, X.H., Cui, R., Yang, C.: Unity3D serious game engine for high fidelity virtual reality training of remotely-operated vehicle pilot. In: 2018 10th International Conference on Modelling, Identification and Control (ICMIC) (2018)
11. Hwang, W.-Y., Hu, S.-S.: Analysis of peer learning behaviors using multiple representations in virtual reality and their impacts on geometry problem solving. Comput. Educ. **62**, 308–319 (2013)
12. Kaufmann, H., Schmalstieg, D., Wagner, M.: Construct3D: a virtual reality application for mathematics and geometry education. Educ. Inf. Technol. **5**(4), 263–276 (2000). https://doi.org/10.1023/A:1012049406877
13. Liu, D., Dede, C., Huang, R., Richards, J. (eds.): Virtual, Augmented, and Mixed Realities in Education. SCI. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5490-7
14. Dijkstra, E.W.: The humble programmer. Commun. ACM **15**(10), 859–866 (1972)
15. Statter, D., Armoni, M.: Teaching abstract thinking in introduction to computer science for 7th graders. In: Proceedings of the 11th Workshop in Primary and Secondary Computing Education, New York, NY, USA, pp. 80–83 (2016)
16. Tanenbaum, A.S., Bos, H.: Modern Operating Systems, 4th edn. Pearson, Boston (2014)
17. Stallings, W.: Operating Systems: Internals and Design Principles, 9th edn. Pearson, New York (2017)
18. Silberschatz, A., Gagne, G., Galvin, P.B.: Operating System Concepts, Enhanced eText, 10th edn (2018)
19. Du, W., Shang, M., Xu, H.: A novel approach for computer security education using Minix instructional operating system. Comput. Secur. **25**(3), 190–200 (2006)
20. Hovemeyer, D., Hollingsworth, J.K., Bhattacharjee, B.: Running on the bare metal with GeekOS. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, New York, NY, USA, pp. 315–319 (2004)
21. Hermocilla, J.A.C.: ICS-OS: a kernel programming approach to teaching operating system concepts. Philippine Inf. Technol. J. **2**(2), 25–30 (2009)
22. Maia, L.P., Machado, F.B., Pacheco Jr, A.C.: 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, New York, NY, USA, pp. 218–222 (2005)
23. Ontko, R., Reeder, A., Tanenbaum, A.S.: Modern Operating Systems Simulators (MOSS). http://www.ontko.com/moss/. Accessed Jan 2020
24. Lopes, Á.R., Souza, D.A., Carvalho, J.R.B., Silva, W.O., Sousa, V.L.P.: SIME: memory simulator for the teaching of operating systems. In: 2012 International Symposium on Computers in Education (SIIE), pp. 1–5 (2012)
25. Jones, D.: RCOS. Java: an animated operating system for computer science education. In: Proceedings of the 1st Conference on Integrating Technology into Computer Science Education, New York, NY, USA, p. 233 (1996)
26. Moreno, L., González, E.J., Popescu, B., Toledo, J., Torres, J., Gonzalez, C.: MNEME: a memory hierarchy simulator for an engineering computer architecture course. Comput. Appl. Eng. Educ. **19**(2), 358–364 (2011)
27. Putchala, M.K., Bryant, A.R.: Synchron-ITS: an interactive tutoring system to teach process synchronization and shared memory concepts in an operating systems course. In: 2016 International Conference on Collaboration Technologies and Systems (CTS), pp. 180–187 (2016)
28. Guimarães, M.P., Scamati, V., Neto, M.P., Martins, V.F., Dias, D.R.C., Brega, J.R.F.: A process-scheduling simulator based on virtual reality technology. In: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1–6 (2016)

29. Smith, M.F.: Software Prototyping: Adoption, Practice, and Management. McGraw-Hill Book Co Ltd, London; New York (1991)
30. Kalmpourtzis, G.: Educational Game Design Fundamentals: A Journey to Creating Intrinsically Motivating Learning Experiences, 1 edn. CRC Press (2018)
31. Unity, Unity. https://unity3d.com
32. Woollard, W.J.: The rôle of metaphor in the teaching of computing; towards a taxonomy of pedagogic content knowledge, Ph.D. University of Southampton (2004)
33. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Trans. Graph. **5**(2), 110–141 (1986)
34. Oberhauser, R., Lecon, C.: Virtual reality flythrough of program code structures. In: Proceedings of the Virtual Reality International Conference - Laval Virtual 2017, New York, NY, USA, pp. 10:1–10:4 (2017)
35. Likert, R.: A technique for the measurement of attitudes. Arch. Psychol. **22**(140), 55 (1932)