


# Learning Operating Systems with Educational Games: A Systematic Mapping of the Literature

Vítor Hugo S. de Camargo  [ State University of Maringá | [viktorhugo99001@gmail.com](mailto:viktorhugo99001@gmail.com) ]  
Maurílio M. Campano Junior   [ State University of Maringá | [maurilio.campanojr@gmail.com](mailto:maurilio.campanojr@gmail.com) ]  
Felippe Fernandes da Silva  [ State University of Maringá | [felippefernandes10@gmail.com](mailto:felippefernandes10@gmail.com) ]  
Linnyer Beatrys Ruiz Aylon  [ State University of Maringá | [lbruiz@uem.br](mailto:lbruiz@uem.br) ]

 Informatics Departament, State University of Maringá, Av. Colombo, 5790, Zona 7, Maringá, PR, 87020-900, Brazil.

Received: 25 February 2025 • Accepted: 21 August 2025 • Published: 25 August 2025

**Abstract:** *Background:* Teaching Operating Systems (OS) is challenging due to the complexity of its concepts, requiring both theoretical understanding and practical application. Educational games have emerged as an engaging strategy to enhance learning, making abstract topics more interactive and accessible. *Purpose:* This study aims to systematically map the literature on educational games used for OS teaching, identifying their main characteristics, covered concepts, and effectiveness. The search was conducted in Google Scholar, IEEE Xplore, SciELO Portal, and CAPES Journals, targeting articles published between 2014 and 2024. *Methods:* The selected studies were analyzed according to research questions related to game content, effectiveness, challenges, and integration into teaching. *Results:* Sixteen educational games were identified, addressing 20 OS concepts. Process management and scheduling were the most covered topics, while mutual exclusion, critical sections, and virtual memory were less explored. All games were tested with students, showing positive learning outcomes. Challenges included errors, accessibility issues, and curricular integration. *Conclusion:* Educational games enhance engagement and learning in OS courses, but there is a need to develop more games covering underrepresented topics and improve their integration into curricula.

**Keywords:** Educational games, Operating systems, Systematic mapping.

## 1 Introduction

In computing education, the challenge of finding methodologies that promote meaningful learning is evident. The field demands not only the transmission of complex theoretical concepts but also the development of practical skills such as problem-solving and logical thinking. Moreover, the rapid technological evolution requires educators to constantly update content and tools, making the teaching process a continuous exercise in adaptation and innovation. In this context, Quirino *et al.* [2017] asserts that learning is a constant necessity for students in academic practice, prompting educators to always seek ways to improve the transmission of knowledge.

Various strategies have been explored to address these challenges in computing education. Among them, active learning methodologies [Moran, 2018] stand out, encouraging students to take a central role in the educational process, such as project-based learning, case studies, and gamification. The use of technological tools, such as simulation devices and virtual environments, has proven effective in complementing teaching and enabling students to experience real-world scenarios in a safe and controlled manner.

In conjunction with these tools, playful approaches have emerged as an efficient way to make learning more engaging and dynamic. One example is the MannaKDT, a technology delivery kit distributed for teaching electronics and robotics [da Silva and Aylon, 2022].

Within the universe of playful approaches, the use of educational games has shown particular promise in computing

education. They offer an engaging way to present complex concepts, transforming learning into an interactive and enjoyable experience, improving the understanding of technical concepts, and promoting critical thinking and problem-solving [Battistella and von Wangenheim, 2016].

In Computer Science, areas such as Computer Architecture and Organization, Human-Computer Interaction, Computer Networks, and Operating Systems are often considered less explored compared to other disciplines [Battistella and von Wangenheim, 2016; Clementino *et al.*, 2022]. In these disciplines, the association of theoretical content with practice has been reported as a contributing factor to student attrition in courses [Fukao *et al.*, 2023].

In particular, the teaching of Operating Systems stands out for the complexity of the concepts involved, which require a deep understanding of both theoretical aspects and practical skills. Consequently, there is a challenge in defining a didactic sequence among its various topics [Maziero, 2002].

The use of simulators is highlighted by Rutten *et al.* [2012] and Hilton and Honey [2011], where the authors emphasize the importance of simulators in the academic context, enhancing students' curiosity and interest by visualizing theoretical concepts through graphical representations or games. Ultimately, this area is essential for developing professionals capable of managing computational resources and interacting between hardware and software.

Thus, this study aims to present a systematic mapping of the literature focused on the use of educational games in teaching Operating Systems (OS), seeking to identify the main characteristics of existing games in the field and how

they are being applied in classrooms.

The remainder of this paper is organized as follows: the next section presents examples of simulators aimed at teaching OS and concepts of educational games. Section 3 outlines the specifications of the systematic mapping, and Section 4 discusses the results obtained. Finally, the conclusions and future work are presented in Section 6.

## 2 Theoretical Foundation

The definition of an operating system can be understood as an extended machine, enabling users to utilize and access a computer's hardware functionalities while managing the computer's existing resources among the processes running on it [Tanenbaum and Bos, 2015; Tanenbaum *et al.*, 1997].

In the study of Operating Systems, one of the main concepts associated with the field is that of processes, which represent an abstraction of a program managed by the OS. On average, a computer manages approximately 130 processes simultaneously, with one process utilizing the processor's resources at a given moment while others wait their turn to execute [Tanenbaum and Bos, 2015].

In addition to managing processes, the operating system oversees memory usage by allocating memory to processes when they are initiated and releasing it when a process is terminated. Moreover, the OS controls which memory regions a process can access, preventing one process from accessing memory allocated to another [Tanenbaum and Bos, 2015].

The management of input and output (I/O) resources is also the responsibility of the OS, which communicates with the I/O module, relaying user-requested tasks and receiving information when needed [Tanenbaum *et al.*, 1997].

The management of these resources can also lead to a deadlock, a problem associated with the concurrent use of two or more resources by processes. Deadlocks must be addressed by the OS, as it manages the processes' use and access to resources [Tanenbaum and Bos, 2015; Machado and Maia, 2004].

Since hard drives (HDDs) and solid-state drives (SSDs) are also input and output devices, the operating system is responsible for managing their usage by processes and users. Thus, the organization and management of files stored on these devices are carried out by the OS, aiming to optimize and facilitate access to the data allocated on them [Tanenbaum and Bos, 2015; Machado and Maia, 2000].

These concepts related to operating systems—such as process and memory management, interprocess communication, I/O management, file systems, and others—are of great importance in the field, as they form the foundation for Computer Networks and Distributed Systems [Coulouris *et al.*, 2013; Kurose and Ross, 2010].

### 2.1 Simulators and Operating System Topics

These operating system topics also serve as a foundation for building compilers for programming languages (PL), as a PL utilizes a computer's computational resources, which are managed by the OS [Sebesta, 2018].

In general, the main operating system concepts include process and memory management, file operations and the file system as a whole, input/output operations management, deadlock handling, as well as multiprocessor system management and security-related process issues.

The teaching of these concepts is often associated with simulators, as the processes within an OS are dynamic and can be represented through simulators or educational games [Maia, 2001]. The author also emphasizes that associating theoretical concepts with practical activities conducted in laboratories is essential. Table 1 presents examples of simulators aimed at teaching operating systems.

**Table 1.** Simulators Focused on Teaching Operating Systems

Simulator	Reference
IO Simulator	Medeiros <i>et al.</i> [2011]
NACHOS	Christopher <i>et al.</i> [1993]
OS Simulator	Gadelha <i>et al.</i> [2010]
RCOS.java	Jones and Newman [2001]
SimulateOS	Freitas [2023]
SOIS	Cruz <i>et al.</i> [2008]
SOsim	Maia [2001]
SSOG	Kioki <i>et al.</i> [2008]
SWSO	Oliveira and Souza [2015]
TBC-SOWeb	Reis and Costa [2009]
WebJuvia	Silva <i>et al.</i> [2021]
WxProc	Rocha <i>et al.</i> [2004]

Medeiros *et al.* [2011] presents the Web IO Simulator tool, aimed at teaching the management of input and output devices performed by the Operating System. The software's goal is to promote accessible and intuitive learning by providing visual methods to understand OS concepts.

Nachos [Christopher *et al.*, 1993] simulates a Unix-like environment, resembling a real computer with a processor, memory, and input/output devices. The tool is instructional, allowing users to modify and test aspects of an operating system's design within a controlled environment.

The OS Simulator Gadelha *et al.* [2010] is designed for teaching OS concepts, focusing on file system management. The simulator enables interaction in a virtual environment where users can create, delete, read, and write files and manipulate directories.

Also emphasizing visual interaction, RCOS.java allows users to engage in a simulated environment, working on concepts such as memory management, scheduling, and process communication [Jones and Newman, 2001].

SimulateOS enables users to assign attributes to processes, visualizing the state of each process in the simulated environment (running, ready, or blocked). Additionally, the tool allows users to adjust the processor clock, the time quantum allocated to each process, and the waiting time for I/O operations [Freitas, 2023].

Associated with computer architecture, SOIS [Cruz *et al.*, 2008] includes a processor simulator integrated with the machine control performed by the operating system. The software enables users to develop, execute, and monitor programs in a strictly controlled environment, supporting education and research in the field.

Maia [2001] describes the SOsim simulator, focused on teaching concepts such as process management, scheduling, memory management, and more. The tool provides real-time interaction and visualization of operations, aiming to enhance the learning experience for this discipline.

The SSOG tool targets students and teachers working on operating systems courses. Its interface allows users to simulate process execution, scheduling algorithms, and memory management. Additionally, the software addresses concepts such as interprocess communication, synchronization, and file systems Kioki *et al.* [2008].

To support learning fundamental operating system concepts, SWSO [Oliveira and Souza, 2015] is an online platform with a user-friendly interface. It enables users to interact with and explore various OS features, simulating different process scheduling algorithms, memory management techniques, and disk access strategies.

Designed to teach scheduling policies and memory allocation, TBC-SO/Web aims to facilitate learning complex concepts related to operating systems. The simulator offers an intuitive interface with interactive visualizations of OS algorithm functionality [Reis and Costa, 2009].

Focusing on memory management performed by the operating system, WebJuvia [Silva *et al.*, 2021] allows users to interact with memory allocation algorithms such as First-Fit, Best-Fit, Worst-Fit, and Next-Fit. The software aims to make these concepts easier to understand through a practical and interactive tool.

To simplify the understanding of scheduling policies, Wx-Proc provides a graphical visualization of classic scheduling algorithms, such as First-Come-First-Served, Round Robin, Shortest-Job-First, and Remaining-Job-First [Rocha *et al.*, 2004].

Based on the information about the simulators described above, Table 2 presents the relationship between OS content and the simulators that address them. Figure 1 illustrates the operating system concepts covered by the simulators found in the literature.

Based on the data in Table 2, Figure 1, and the description of the simulators, it is evident that the most covered concepts in the simulators are memory management and process management and scheduling.

On the other hand, concepts such as interrupts, input/output management, file systems, synchronization, interprocess communication, virtual memory, threads, and disk scheduling and management are among the least covered in the described simulators.

## 2.2 Educational Games and Their Core Components

As described in Maia [2001], associating these concepts with simulators and/or educational games can be a way to engage and motivate students during the teaching-learning process.

Digital educational games can enhance the formation of concepts, content, and skills embedded in the game. Additionally, they provide an imaginary context to be explored and are motivating due to promoting challenges, fantasy, and curiosity [Falkembach *et al.*, 2006].

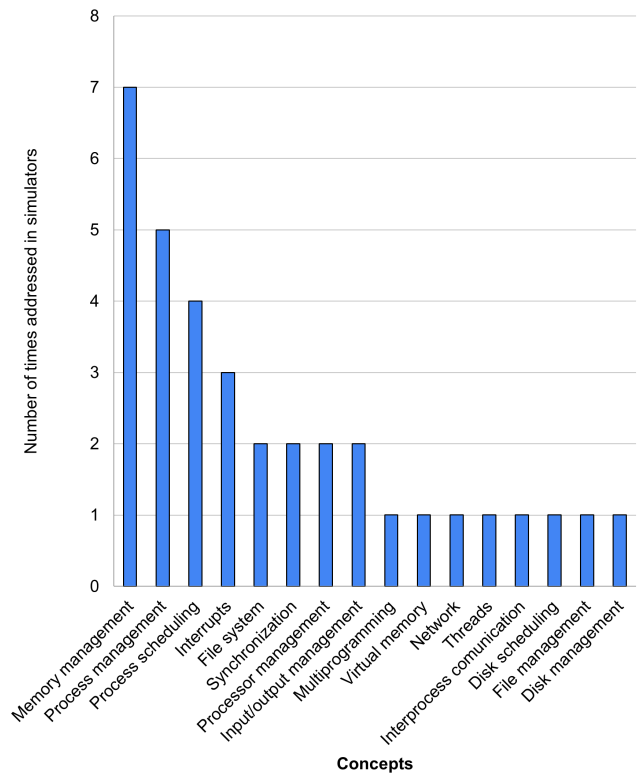


Figure 1. Operating System Concepts Covered by the Simulators Found

Costa [2009], define the characteristics of an educational game as having a structure similar to a knowledge object, recognizable by the player, where understanding this structure is essential to achieving success in the game. Furthermore, the authors emphasize that everything in an educational game should focus on fun and entertainment.

Moreover, a game is composed of four basic elements: goal, rules, feedback, and voluntary participation. Goals define the actions the player must take to achieve the game's main objective, while rules set the limitations imposed on the user. Feedback guides the player based on correct or incorrect actions, and voluntary participation ensures the player decides whether or not to play [Pascotini, 2018].

However, an educational game must balance challenge with player motivation, as this balance is crucial for learning, referred to as Flow [Sanches, 2019].

In Computer Science, educational games have been widely used in various areas [Battistella and von Wangenheim, 2016; Clementino *et al.*, 2022]. In fields like Software Engineering and Programming, numerous games can be found in the literature, as discussed in works like Haddad *et al.* [2024] and de Barros Costa and Rocha [2018].

In Haddad *et al.* [2024], 43 games aimed at teaching software engineering are presented, while de Barros Costa and Rocha [2018] reports the existence of 30 games for teaching programming.

In areas like Digital Circuits (DC), Data Structures (DS), and Formal Languages and Automata (FLA), fewer games are found in the literature, as studies by Santini *et al.* [2023], Julio *et al.* [2024], and Santini *et al.* [2022] present 13, 16, and 9 games respectively for DC, DS, and FLA.

The number of games identified in the above studies reflects the observation by Battistella and von Wangenheim [2016] and Clementino *et al.* [2022], which points out that

	IO Simulator	NACHOS	OS Simulator	RCOS.java	SimulateOS	SOIS	SOSim	SSOG	SWSO	TCB-SO/WEB	WebJuvia	wxProc
Interprocess Communication				X								
Disk Scheduling				X								
Process Scheduling				X				X		X		X
File Management									X			
Disk Management									X			
Input/Output Management	X					X						
Memory Management				X		X	X	X	X	X	X	
Process Management					X	X	X	X	X			
Processor Management							X	X				
Interrupts				X		X						
Virtual Memory		X										
Multiprogramming		X										
Networking		X										
Synchronization		X						X				
File System			X					X				
Threads		X										

**Table 2.** Operating System Topics Associated with Simulators Found in the Literature

areas like programming, software engineering, and security have a larger number of games compared to other fields in Computer Science.

Regarding the area of Operating Systems, Battistella and von Wangenheim [2016] presents only three examples of games, while Clementino *et al.* [2022] found no games associated with OS concepts.

In addition to the mentioned games and simulators, an expanded search in scientific databases and digital repositories may reveal other games aimed at teaching operating systems concepts. Identifying new games can provide additional resources to complement the teaching of operating systems, promoting greater engagement and understanding among students.

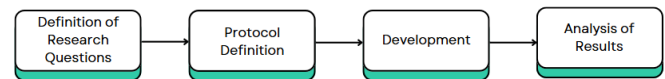
Despite their utility, the analyzed simulators show gaps in the range of covered content. Concepts such as inter-process communication, threads, virtual memory, disk management and scheduling, and aspects related to file systems appear less frequently in the identified tools. Including these topics in new simulators or expanding existing ones could offer a more comprehensive learning experience aligned with the needs of modern teaching.

In conclusion, simulators and educational games play a vital role in teaching operating systems. They combine theoretical concepts with interactive practices, aiding in content retention and skill development. Continuous study and improvement of these tools, along with the search for new resources, are essential to ensure robust and comprehensive training in operating systems, contributing to the advancement of teaching and research in Computer Science.

### 3 Methodology

This study expands upon the work presented in de Camargo *et al.* [2024], incorporating new questions into the systematic mapping and providing a more in-depth analysis of the research conducted by the authors.

The methodology of this work was adapted from the model described by Kitchenham and Brereton [2013] and presents a systematic mapping of educational games for Operating Systems instruction. The stages of this model can be observed in Figure 2.



**Figure 2.** Methodology adopted in this work

#### 3.1 Definition of Research Questions

The objective of this work is to explore educational practices that use games in teaching operating systems, their effectiveness, the types of games used, as well as other relevant characteristics for the teaching-learning process. Thus, the following research questions were defined:

- **Q1:** Which operating system concepts do the games address?
- **Q2:** Were the games used with students? What was the number of students involved?
- **Q3:** How were the tests conducted, and what were the results?
- **Q4:** What challenges were encountered in the application of the games?
- **Q5:** How were the games incorporated into teaching Operating Systems?
- **Q6:** Were the games used to teach new concepts or to reinforce existing concepts in operating systems?
- **Q7:** What are the genres of the games?
- **Q8:** What platforms are the games available on?
- **Q9:** In which programming languages are the games developed?
- **Q10:** Which educational games are currently available?

#### 3.2 Protocol Definition

The protocol definition involves creating the search string, determining the inclusion and exclusion criteria, and selecting the search sources to be used. These items are described below.

##### 3.2.1 Definition of the Search String

Based on the definition of the research questions, a search string was developed to identify relevant studies. This string is represented as: "S1 AND S2 AND S3 AND S4," where each term "S" in the string can be defined as follows:

- **S1:** serious game OR teaching game OR learning game OR educative game OR educational game OR game-based;

- **S2:** operating system OR operating systems;
- **S3:** course OR concepts OR curriculum OR subject OR area of OR fundamentals OR algorithms OR module;
- **S4:** race condition OR threads OR process scheduling OR semaphores OR deadlock OR mutex OR bus OR system calls OR inter-process communication OR busy waiting OR strict alternation OR mutual exclusion OR memory management OR paging OR critical region OR lock variables OR monitor OR concurrent OR parallel.

This search string reflects a systematic approach to investigating the application of educational games in operating systems education, encompassing both broad terms (e.g., serious games and game-based learning) and domain-specific topics. The intersection between the enumerated search strings ensures a targeted search while maintaining the relevance of identified studies.

It is worth noting that multiple search strings were tested across databases to identify the most effective formulation. The string was calibrated based on an analysis of the precision of the results obtained during exploratory testing. Initially, the string included only terms related to games and operating systems, but the results showed low precision, returning many irrelevant articles. To refine the search, a second version of the string was developed by adding terms related to education and teaching. However, many of the returned articles still focused on educational games in areas unrelated to operating systems. To improve precision, a consultation was held with a professor of the Operating Systems course, who assisted in identifying key concepts in the field. These specific terms were then incorporated into the final version of the string, resulting in a more effective filtering of relevant studies, as presented above.

### 3.2.2 Criteria

Once the research questions and the search string were developed, inclusion and exclusion criteria were defined to select the articles relevant to the study. These criteria are presented in Tables 3 e 4.

**Table 3.** Defined Inclusion Criteria

	<b>Inclusion Criteria</b>
<b>CI1</b>	The study should consider educational games to support Operating Systems instruction.
<b>CI2</b>	The paper addresses concepts related to operating systems.

**Table 4.** Defined Exclusion Criteria

	<b>Exclusion Criteria</b>
<b>CE1</b>	The article does not satisfy the search string
<b>CE2</b>	The work was published more than 10 years ago
<b>CE3</b>	The work does not have a complete version available for download
<b>CE4</b>	The work is not freely accessible
<b>CE5</b>	The work is written in languages other than English and Portuguese

Given the rapid technological advancements and the growing proliferation of digital games in recent years, this study

prioritizes scientific publications aligned with the current context. For this reason, a timeframe was established focusing on articles published within the last decade.

### 3.2.3 Search Sources

The search sources used in this study include **Google Scholar**, **IEEEExplore**, **SciELO**, **SBC-OpenLib** and **CAPES Journal database**. These sources were selected as they encompass relevant databases in the field of computing research. It is noteworthy that the searches were conducted exclusively with the search string in English, considering that articles in Portuguese also provide an abstract containing keywords related to the work. Databases such as Scopus and the ACM Digital Library were not included in this research, as Google Scholar already indexes publications from various sources, including part of the content from these platforms. This approach aimed to avoid duplication of results and make the screening process more efficient.

Another relevant aspect to highlight is that the search string needed to include specific terms related to operating systems. This is because a search for generic terms, such as "educational game operating systems," could result in articles about educational games targeted at other areas, which merely mention that the game is intended for operating system X or Y, significantly increasing the number of results returned.

## 3.3 Development

The development of this work was carried out in three main stages, ensuring a rigorous and systematic analysis of the selected literature, namely: (i) Initial Screening; (ii) Abstract-Based Selection; and (iii) Full Evaluation. A complete description of the adopted methodological protocol, including inclusion and exclusion criteria, search string, consulted databases, and data extraction process, is available at the following link<sup>1</sup>.

**Initial Screening:** In this stage, the titles of the papers were analyzed to assess their relevance to the topic of educational games applied to the teaching of operating systems. Papers whose titles were not relevant to the topic were excluded.

**Abstract-Based Selection:** After the initial screening of the studies, the abstracts were carefully analyzed to verify their alignment with the previously established inclusion criteria. Thus, the studies that did not meet the defined criteria or were deemed irrelevant to the research questions were discarded at this stage.

**Full Evaluation:** The studies not excluded in the previous stage were analyzed in their entirety, allowing for a thorough evaluation of the relevance of each study in relation to the inclusion and exclusion criteria defined during the planning phase.

<sup>1</sup>[https://drive.google.com/drive/folders/1fkIqdk12VyRbEI6Lkwhpf1Rmws\\_Fe2h0](https://drive.google.com/drive/folders/1fkIqdk12VyRbEI6Lkwhpf1Rmws_Fe2h0)



### 3.4 Analysis of Results

The analyzed results followed a qualitative approach, in which the studies were organized in a table containing information such as the name of the game, a brief description, and their respective references. Subsequently, each study was analyzed in detail in relation to the research questions, aiming to provide well-founded answers to these questions. Accordingly, the responses to each research question are presented and discussed in detail in Section 4.

## 4 Results

After defining the relevant points of the mapping, the search was conducted during the months of October and November 2023, and the results can be found in Table 5.

**Table 5.** Results of the search and filtering from stages 1, 2, and 3

Base de dados	Stage 1	Stage 2	Stage 3
Google Scholar	373	31	14
IEEEExplore	5	2	1
Scielo	0	0	0
SBC-OpenLib	0	0	0
CAPES Journal database	3	1	0
<b>Total</b>	<b>381</b>	<b>34</b>	<b>15</b>

Based on the initial results of the search (step 1), the second step aimed to select the articles through the analysis of the title and abstract. In the third step, the articles were fully analyzed, resulting in the selection of 15 relevant articles.

It is important to highlight that each step was performed and validated individually by the authors. The works selected by consensus advanced to the next phase, while those that raised doubts were evaluated collaboratively.

Finally, employing the snowballing technique, additional searches were conducted through the references of identified articles, resulting in the inclusion of one additional relevant study. This brought the total number of selected works to 16. These games are presented in Table 6.

Figure 3 presents the screens of the games Parallel [Zhu et al., 2019] and Parallel Islands [Cameron, 2023], while Figure 4 describes the games Speed Schedule [Figueiredo et al., 2020] and Mutual Exclusion Social Game [Popović et al., 2018]. The figures above show that the identified games present distinct graphical and gameplay characteristics. Additionally, Table 6 provides a brief description of each of the games found.

The examined games feature playful and intuitive interfaces specifically designed to teach operating systems concepts. Each game employs distinct mechanics and interaction paradigms. The specific results for each of the 10 research questions defined are presented below.

#### 4.1 Q1: What Operating Systems concepts do the games address?

Among the 16 games found, 20 different concepts associated with Operating Systems are addressed. These concepts can



**Figure 3.** Screenshots of the games Parallel [Zhu et al., 2019] and Parallel Islands [Cameron, 2023]

be seen in Figure 5. It is worth noting that a single game may present more than one concept at the same time.

Aiming to compare the most addressed concepts in simulators (Figure 1) with the concepts covered in games (Figure 5), it is observed that process management concepts stand out in both games and simulators. The dynamism and interactivity associated with the content favor the use of games and simulators in the teaching process.

These concepts represent fundamental topics in the discipline, yet they remain underrepresented in current educational tools. While various pedagogical approaches could enhance student comprehension, key areas such as mutual exclusion, critical sections, monitors, threads, and virtual memory receive minimal coverage in existing games.

Notably, among these concepts, only threads are addressed in simulator-based approaches—and even then, by just a single implementation. It is also noteworthy that concepts such as deadlock, input/output management, and security issues were not addressed in the educational games identified.

#### 4.2 Q2: Were the games used with students? What was the number of students involved?

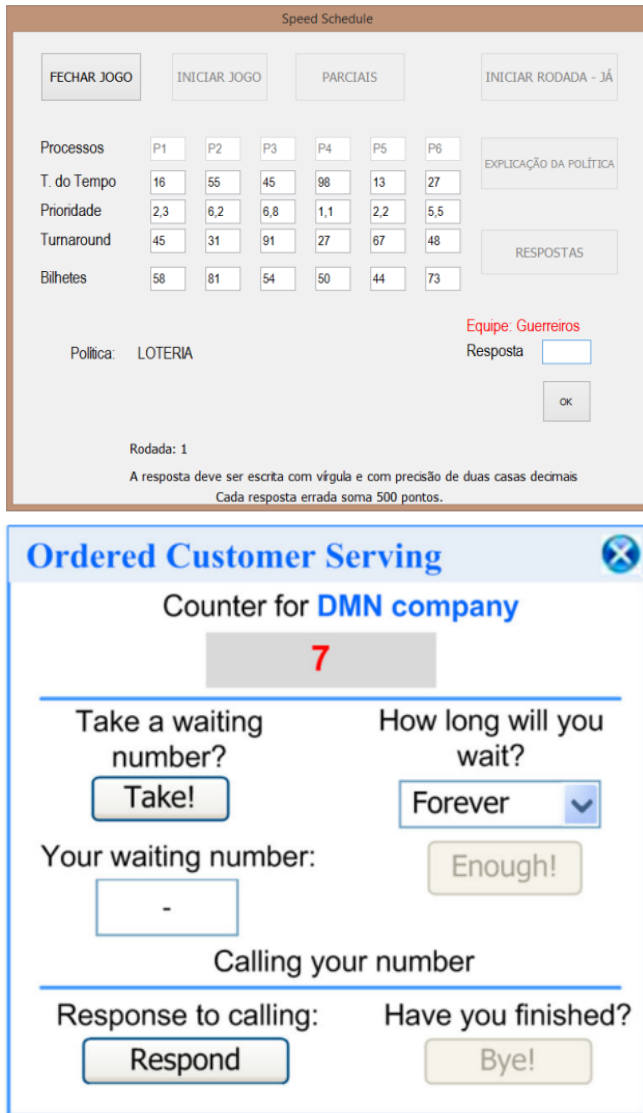
Among the 16 games found, all were tested with students. Regarding the number of students involved in the tests, the game that was tested with the most students was Operating System Little Pet [She et al., 2013], with 141 students participating in the test.

On the other hand, the game Parallel Islands [Cameron, 2023] presented test results with only 8 students. Information on the other games can be seen in Table 7.

It is also worth noting that the game Speed Schedule did not indicate the number of students, only reporting that three

**Table 6.** Educational games for Operating Systems

Game Name	Description	Reference
Arena deadlock	Board game developed to teach deadlock concepts. The game simulates different situations where processes are waiting due to resources held by other processes	de Jesus Santos <i>et al.</i> [2020]
Escalonando	Game that addresses topics such as process scheduling, process states, and other Operating Systems topics	Luccas [2019]
Mobile Virtual Reality Game-based learning	Virtual reality game designed to teach programming algorithms used in Operating Systems courses	AbdelAziz <i>et al.</i> [2020]
Mutual Exclusion Social Game	Educational game that simulates mutual exclusion problems. Students use the "Geppeto" widget to ensure they coordinate adding income to a virtual account simultaneously	Popović <i>et al.</i> [2018]
OpenTTD	Open-source version of the game "Transport Tycoon Deluxe." It is a simulation game for building and managing computer networks	Murphie and Hansen [2018]
Operating System Little Pet	Game integrated into <i>Facebook</i> where students take care of and strengthen their pets by answering questions about the content of the Operating Systems course	She <i>et al.</i> [2013]
Parallel	Educational game that teaches concurrent and parallel programming concepts through challenges where players must employ different problem-solving strategies	Zhu <i>et al.</i> [2019]
Parallel Islands	Users must travel between islands, obtaining provisions for their survival by completing tasks related to parallel programming	Cameron [2023]
Prototype on virtual memory	Platform game where the character faces challenges such as translating virtual addresses by applying virtual memory concepts	Souza [2014]
Race condition	This game addresses topics such as inter-process communication, race conditions, mutual exclusion, and other important Operating Systems topics	Luccas [2019]
No name	Users interact with characters that must be programmed to perform tasks by applying parallel programming concepts	DeLozier and Shey [2023]
Space Shooter	Interactive web game designed to help students understand the concepts associated with <i>buffer overflow</i>	Zhang <i>et al.</i> [2020]
Speed Schedule	Game that addresses scheduling policies in Operating Systems, offering players the opportunity to learn about process ordering and average turnaround time calculation interactively	Figueiredo <i>et al.</i> [2020]
Temple of Treasures	Game that addresses Operating Systems concepts related to access control, enabling players to manage permissions and access rights to computer devices and resources.	Weanquoi <i>et al.</i> [2021]
Threadman	Game that addresses topics such as process scheduling, process states, and other Operating Systems topics	Luccas [2019]
World of Operating System (WoUSO)	Browser-based game designed to complement an introductory Operating Systems course involving quizzes, puzzles, challenges, and competitions	Rughiniş [2013]



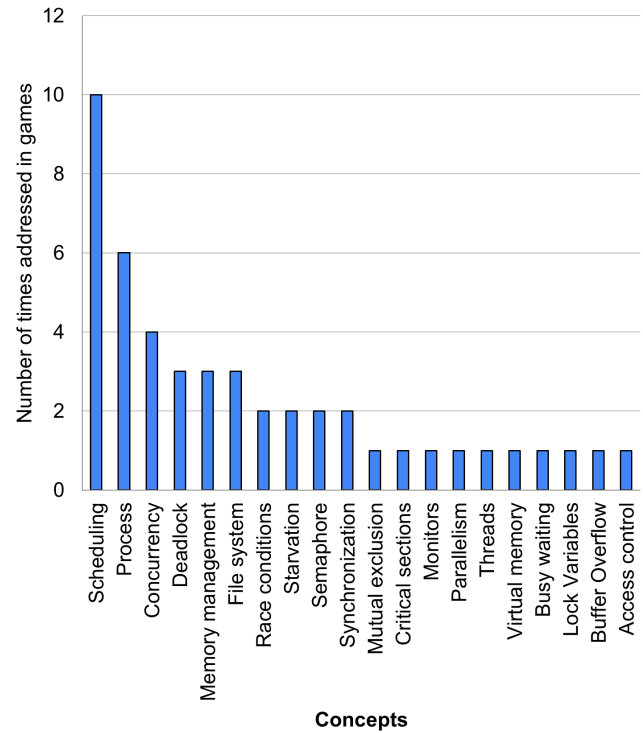
**Figure 4.** Screenshots of the games Speed Schedule [Figueiredo *et al.*, 2020] and Mutual Exclusion Social Game [Popović *et al.*, 2018]

Operating Systems classes participated in the tests.

### 4.3 Q3: How were the tests conducted, and what results were obtained?

Among the evaluations and tests carried out with the identified games, recurring methods include: pre- and post-test questionnaires [She *et al.*, 2013; Zhu *et al.*, 2019; Cameron, 2023; Zhang *et al.*, 2020; Weanquoi *et al.*, 2021], questionnaires based on specific game evaluation methodologies [Luccas, 2019; de Jesus Santos *et al.*, 2020; Souza, 2014; AbdelAziz *et al.*, 2020], player observations [Figueiredo *et al.*, 2020; Rughiniş, 2013], use of control groups [Popović *et al.*, 2018], questionnaires focused on the concepts explored [Murphie and Hansen, 2018], and validation of student-proposed solutions against predefined correct solutions [DeLozier and Shey, 2023]. Detailed information about each game evaluation is presented below.

The evaluation of the Arena Deadlock game, aimed at assessing students' understanding of deadlock, was conducted after a brief conceptual introduction and explanation of the game rules. The evaluative questionnaire included 10 ques-



**Figure 5.** Operating Systems concepts addressed in the educational games found

**Table 7.** Number of students per educational game

Game Name	Number of Students
Arena deadlock	17
Escalonando	20
Mobile Virtual Reality Game-based learning	110
Mutual Exclusion Social Game	25
OpenTTD	16
Operating System Little Pet	141
Parallel	25
Parallel Islands	8
Prototype on virtual memory	10
Race condition	20
No name	95
Space Shooter	41
Speed Schedule	three OS classes
Temple of Treasures	53
Threadman	20
World of Operating System (WoUSO)	118

tions about OS concepts and the game itself [de Jesus Santos *et al.*, 2020].

The evaluation results indicate that the game effectively conveys the concept of deadlock, facilitating understanding. Students suggested playing without prior deadlock concepts to encourage learning during gameplay by establishing connections with their in-game experiences.

Escalonando, Race Condition, and Threadman were created by the same author [Luccas, 2019] and evaluated together using common questions for all three games. The



evaluation was divided into four stages. In the first, students expressed opinions about the games, focusing on fun, intuitiveness, content clarity, and playfulness.

Results from this stage showed that 90% of students considered the games fun and intuitive, with clear and objective content, and 95% felt the games contributed to their learning.

In the second stage, students associated the games with the concepts covered in the operating systems course, with the majority correctly linking the game content to the course material.

To assess concept learning more deeply, the third stage included essay questions where students described the characteristics of algorithms used in the games, their advantages, and disadvantages in an OS. Most students responded correctly with detailed answers, demonstrating good understanding.

Finally, in the fourth stage, students provided feedback on the games, suggesting improvements to the aesthetics and mechanics and recommending other OS topics for inclusion. They also reported some bugs in the games.

The Mobile Virtual Reality Game-based Learning [Abdelaziz *et al.*, 2020] tests aimed to assess the tool's ability to motivate students to learn about scheduling algorithms in operating systems.

The study spanned six weeks. In the first week, students studied scheduling algorithm concepts. In the second, before using the tool, students completed the Instructional Materials Motivation Survey (IMMS) to measure four motivation components: attention, relevance, confidence, and satisfaction.

In the fourth and fifth weeks, students participated in learning activities related to processor programming in two distinct environments.

In the final week, students retook the IMMS to evaluate motivation changes. Results showed significant increases in confidence and satisfaction, while attention and relevance remained unchanged.

In the Mutual Exclusion Social Game [Popović *et al.*, 2018], the evaluation aimed to examine the effectiveness of an experimental approach to mutual exclusion in parallel programming. The 25 students involved in this evaluation phase were divided into two groups, one with 11 students forming the conventional sample and another with 14 students as the experimental sample.

The conventional group used traditional methodology, classroom theory, and supporting bibliographic references for the discipline. Meanwhile, the experimental group used the proposed tool during the course.

In the end, both groups were subjected to three distinct tasks related to programming in C language and the use of mutual exclusion. In the conventional group, 7 out of 11 students solved the problems, resulting in a success rate of 64%. In contrast, the experimental group achieved a task success rate of 79%, representing a 15% increase over the conventional group.

For the evaluation of OpenTTD [Murphie and Hansen, 2018], students were first familiarized with the game, aiming to learn how to navigate the interface and understand the resources necessary for the subsequent tasks.

Students then solved programming problems related to concurrency, involving race conditions and deadlock. Sub-

sequently, the questions used aimed to associate the game's concepts with the contents addressed in the programming task.

The feedback from the students was notably positive, with several students reporting that the activities helped clarify and solidify abstract concepts. Additionally, student engagement was evident from the fact that even after completing the tests with the tool, some students continued to interact with the game voluntarily to practice and learn.

The evaluation of Operating System Little Pet [She *et al.*, 2013] also divided students into an experimental group (70 students) and a control group (71 students). All students answered a quiz on prior OS knowledge, in addition to a questionnaire aimed at understanding their motivation for learning the course content.

Both groups used the game; however, the experimental group used a version with social interactions between the game's characters, while the control group's version did not allow social interaction among players.

After the activities with the game, the authors observed that the experimental group's motivation increased, whereas the control group's motivation decreased. A new questionnaire was administered to the students, assessing the game's impact on their motivation and understanding of the subject matter.

Based on the results, the authors stated that the tool was helpful in improving the course's comprehension, even with feedback indicating areas for improvement and issues with the game.

After using the game Parallel [Zhu *et al.*, 2019], students were challenged to draw connections between the game's puzzles and the concepts covered in theoretical classes. Regarding content related to mutual exclusion and critical sections, 92.3% of the students recognized and reported such connections.

The authors also noted that a qualitative analysis of the students' responses indicated a high level of precision in the answers. Regarding the game itself, students reported that it provided an interactive and dynamic approach to a complex theoretical topic.

For the evaluation of Parallel Islands [Cameron, 2023], initial information was collected about the participants' prior knowledge (pre-game survey). Subsequently, participants played the game and completed a new questionnaire (post-game survey) regarding the knowledge acquired, along with providing feedback on their experience with the game.

The author reports that the success rate on the first questionnaire was 50%, whereas the second achieved 100% success. Students also indicated that the software is intuitive and efficient as a learning tool.

All students involved in the evaluation of the Prototype on Virtual Memory [Souza, 2014] had already completed the OS course. Thus, the author's evaluation focused on the game itself.

Using the Likert scale, the game's usefulness as an educational tool received an excellent rating. As for criteria like organization and clarity of content, the feedback ranged between good and excellent. Finally, the interface was rated as intuitive and attractive, though it received suggestions for improvement from the students.

Another noteworthy point in Souza [2014]’s work was the use of a series of dichotomous questions, i.e., binary answers such as yes or no, true or false. There was unanimity in the responses to these questions on topics such as understanding the subject addressed in the game, the completeness of the content, and the game’s contribution to teaching virtual memory. Students also unanimously indicated that they were not aware of any other educational game with a similar approach.

Finally, the subjective questions revealed positive perceptions from students, with suggestions for improvements to the interface, new characters, online rankings, multiplayer capabilities, and incorporating new content related to operating system memory management.

DeLozier and Shey [2023] report that the evaluation of the proposed game was based on the efficiency of the characters’ movements compared to a pre-established optimized solution.

The results of the study indicate that while participants initially faced challenges adapting to the visual programming language proposed by the game, with time and practice, the vast majority were able to solve complex programming problems involving synchronization.

The Space Shooter [Zhang *et al.*, 2020] was evaluated at two universities, where students initially answered a questionnaire about the concepts involved in the game. After each game module, students completed a quiz, immediately assessing users’ perceptions.

At the end, a new questionnaire on the content was applied alongside a satisfaction survey regarding learning through the game. The results showed an overall increase in students’ performance as they progressed through the game modules, demonstrating the tool’s effectiveness in enhancing comprehension.

The satisfaction survey results at one university indicated that 90.5% agreed that the learning objectives were achieved, while at the other university, the rate was 83.3%. The learning assessment showed a slight improvement between the initial and post-game surveys.

The Speed Schedule [Figueiredo *et al.*, 2020] was evaluated with three operating systems classes at a university, where the game was integrated as a practical learning tool after theoretical explanations in the classroom. Students were grouped into teams of three for sessions of approximately one and a half hours.

The authors report that, initially, the groups quickly dispersed; however, as the game progressed, student engagement and motivation increased. Another point highlighted was that in the early sessions, students consulted theoretical materials to assist in solving tasks, a practice that diminished over time.

The authors also noted that the ranking system and the possibility of breaking other groups’ records stimulated and encouraged learning through the game.

At the end of the game sessions, a test validated the knowledge of both the students who played and those who did not, showing a 14% better performance for the group that used the tool. Additionally, 89% of students rated the game as fun or very fun.

To evaluate the effectiveness and impact of the game Temple of Treasures [Weanquoi *et al.*, 2021], the authors applied

a pre-test and a post-test containing questions related to the game’s content.

In addition to these, in-game questionnaires were presented to users at the end of each game level. The authors also measured player experience, satisfaction, and perceptions of the game.

The results presented indicated an increase in knowledge about operating systems, as well as positive feedback on the game’s quality as an educational tool. Furthermore, the success rate at each game level was used as an evaluation criterion for the content covered by the game, also showing learning gains compared to the pre-test.

Rughiniş [2013] evaluated World of Operating System (WoUSO) using three distinct methods: direct observation of students interacting with the game in multiple sessions, an anonymous survey after gameplay, and an interview.

The evaluation results indicated that the majority of students considered WoUSO a tool that contributed to understanding concepts associated with the course. A noteworthy aspect highlighted by the author was the social component of the game, where players’ rankings and reputations motivated and engaged competition among them.

At the end of the tests, it was concluded that WoUSO proved to be an innovative and effective resource in the educational field, promoting a fun and socially engaging learning environment.

#### 4.4 Q4: What are the difficulties encountered in the application of the games?

Among the difficulties mentioned in the articles, the most frequently discussed factor is the bugs found in the games. Issues such as repetitive music, inability to close tutorials, problems with the game’s save function, and confusing mechanics fall under this category.

One reason that may contribute to the prevalence of bugs in the games is the programmer’s inexperience. Since some educational games are developed by undergraduate or graduate students, this may account for the number of issues identified in the games.

Another significant factor highlighted in three games was the issue with the game’s color palette. Feedback from a colorblind student revealed that they could not distinguish game elements because the colors were not designed with colorblind users in mind.

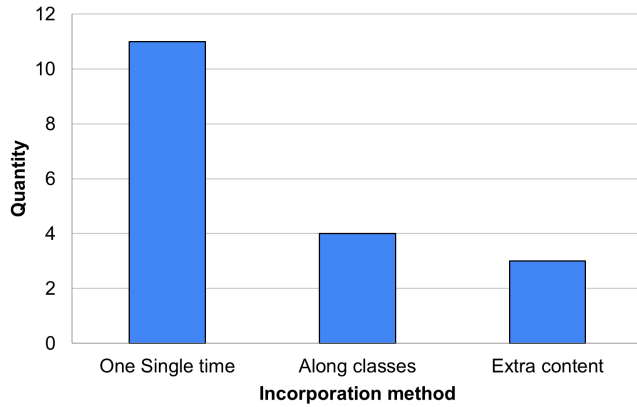
Difficulties related to the content itself were also reported. As the games deal with complex concepts, associating the theoretical content with the practical elements embedded in the game can pose additional challenges.

It is also worth noting problems related to student motivation, the short duration of activities, and the need for specialized hardware to perform certain tasks.

#### 4.5 Q5: How were the games incorporated into the teaching of Operating Systems?

The use of educational games for Operating Systems in the classroom can be classified into three types: used only once,

used throughout the course (more than once), and as extracurricular content. Figure 6 shows how the games were incorporated, with a strong tendency to use the games only once (11 games).



**Figure 6.** Classroom implementation frequency of operating systems educational games

Figure 6 also indicates that most of the games found are used as isolated episodes in classes, underestimating their potential to continuously and cumulatively reinforce concepts.

#### 4.6 Q6: Were the games used to teach new concepts or to consolidate concepts in operating systems?

Regarding how the game was approached in the classroom, 76.2% of the games were used for the consolidation of concepts already covered in class, meaning the students already had prior knowledge of the content addressed in the game.

On the other hand, 23.8% of the games were used at a time when students had no prior knowledge of the concepts associated with the game. These games were specifically designed to introduce core operating systems concepts to novice learners, serving as foundational pedagogical tools.

Although most of the games identified were used after students had been introduced to the content, the above data demonstrates the flexibility of educational games, allowing both the introduction of new concepts and the consolidation of existing ones.

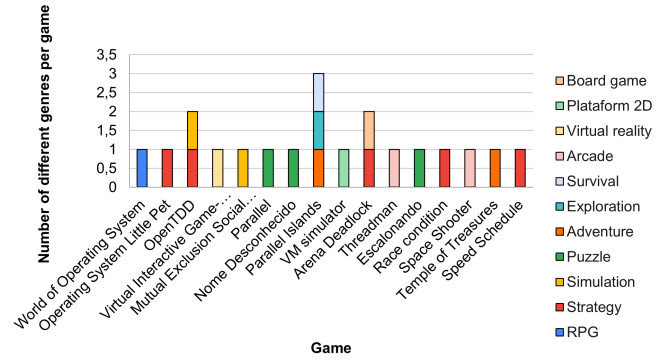
#### 4.7 Q7: What are the genres of the games?

The genres of the games identified can be seen in Figures 7 and 8 below. On the left, the individual genres of the games are presented, noting that a game may fall into more than one genre. On the right, the distribution of the identified genres is shown.

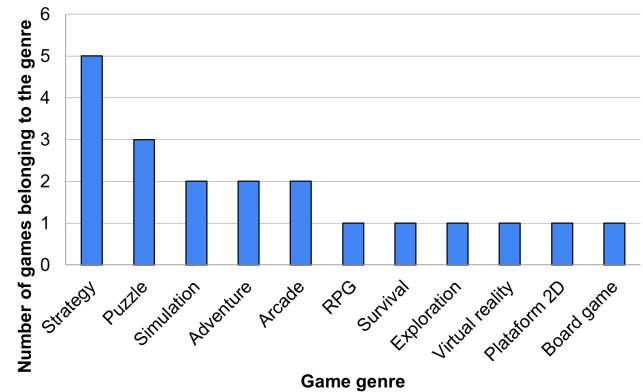
Figure 8 shows the predominance of strategy and puzzle games. This is evident as these types of games rely on problem-solving and critical thinking—essential skills in various disciplines associated with Computer Science.

#### 4.8 Q8: What are the platforms of the games?

Among the identified games, Arena Deadlock [de Jesus Santos et al., 2020] is a board game and Mobile Virtual Reality



**Figure 7.** Genres involved in each identified game



**Figure 8.** Distribution of the identified genres

Game-based learning [AbdelAziz et al., 2020] is designed for mobile devices, as it utilizes virtual reality.

While Arena Deadlock emphasizes hands-on experience with resource allocation scenarios, Mobile Virtual Reality Game-based learning immerses students in 3D environments to visualize abstract OS concepts, demonstrating the diversity of pedagogical approaches across physical and digital platforms. The remaining games (14 instances) were developed for use in web browsers or as standalone computer software.

#### 4.9 Q9: In which programming languages are the games developed?

Only 9 out of the 16 identified games mentioned the programming language used in their development. Among these, 6 were developed in C#: Mobile Virtual Reality Game-based Learning, Space Shooter, Temple of Treasures, Threadman, Escalonando, and Race Condition. Only one game was developed using C and C++ (OpenTTD), one game used JavaScript and Java (Parallel Islands), and one was developed in Delphi/Object Pascal (Speed Schedule). Table 8 describe the languages for all the games founded.

The predominance of C# reflects the use of Unity [Unity, 2024], a popular game development engine. It is also noteworthy that the analyzed studies did not report the use of additional libraries and tools, only listing the programming language employed in the game's development.

**Table 8.** Programming language of games founded

Game Name	Programming language
Arena deadlock	board game
Escalonando	C#
Mobile Virtual Reality	C#
Game-based learning	
Mutual Exclusion Social Game	not especificed
OpenTTD	C and C++
Operating System Little Pet	not especificed
Parallel	not especificed
Parallel Islands	JavaScript and Java
Prototype on virtual memory	not especificed
Race condition	C#
No name	not especificed
Space Shooter	C#
Speed Schedule	Delphi/Object Pascal
Temple of Treasures	C#
Threadman	C#
World of Operating System (WoUSO)	not especificed

#### 4.10 Q10: Which educational games are currently available?

Among the 16 games analyzed, only three are currently available to be played: Temple of Treasures<sup>2</sup> [Weanquoi *et al.*, 2021], WoUSO<sup>3</sup> [Rughiniş, 2013], and OpenTTD<sup>4</sup> [Murphie and Hansen, 2018]. On the other hand, Arena Deadlock [de Jesus Santos *et al.*, 2020], due to its nature as a board game, provides its components, including the board, cards, and items, in digital format for printing and use. This distinction highlights the diversity of formats and accessibility of the games analyzed, with variations ranging from complete digital experiences to physical materials that require user assembly.

Most of the evaluated games are no longer available, possibly due to their discontinuation in the educational context, as many of them were developed for specific, one-time activities. This discontinuity may occur because institutional emails are typically used for research activities. When authors leave the institution, their email accounts are deactivated. A concrete example involves storing games on Google Drive - when researchers depart from the institution, they lose access, and consequently, the links to their previously stored work become unavailable.

## 5 Limitations

Although this study followed a systematic and structured methodology, some limitations may influence the interpretation and generalization of the results. To provide a clearer understanding of these limitations, we discuss the main threats to validity using four widely recognized categories in empir-

ical research: construct validity, internal validity, external validity, and conclusion validity. Each of these dimensions highlights specific aspects of the research process that may impact the reliability and robustness of the findings.

**Construct validity** concerns whether the study effectively captures what it intends to measure. In this mapping, the string of search terms and inclusion criteria were designed to identify educational games for teaching operating systems. However, relevant studies might have been missed due to differences in terminology, vague descriptions, or limited indexing in the selected databases. Furthermore, the definition of what constitutes an "educational game" or a "game focused on operating systems" may vary across studies, introducing subjectivity in the selection and classification processes. Although calibration was carried out iteratively with domain experts, the manual interpretation of game descriptions, mechanics, and educational intent may still introduce bias.

**Internal validity** refers to the extent to which the categorization and interpretation of the data are trustworthy. Since the classification of genres, platforms, programming languages, and evaluation methods involved interpretative judgments, there is a risk of misclassification or inconsistency. In addition, the fact that all 16 identified games were tested with students may suggest a higher degree of maturity or validation. However, the lack of standardized evaluation frameworks across the studies limits the ability to directly compare educational effectiveness or methodological rigor.

**External validity** relates to the generalizability of the findings. This mapping focused exclusively on educational games for operating systems, which restricts the applicability of the results to other Computer Science topics. Moreover, although a variety of genres was identified and most games were digital, the sample size is relatively small (16 games), and cultural or institutional differences may affect their use in other contexts. For example, only one non-digital game (a board game) was found, possibly indicating a publication or development bias toward digital formats.

**Conclusion validity** concerns whether the conclusions drawn are reasonable based on the collected evidence. The descriptive nature of a systematic mapping limits inferential analysis. Furthermore, some games were no longer available for testing, and many studies lacked detailed documentation on game design, deployment, or learning outcomes. This hinders deeper assessments of effectiveness or replication of the educational approaches. As a result, while trends and gaps were identified, caution must be taken not to overgeneralize the implications of individual studies.

## 6 Conclusion

This paper presented a systematic mapping of educational games aimed at teaching operating systems. Among the games identified, all were tested with students and demonstrated positive results as tools to support the teaching and learning of operating systems. A significant portion of the students involved provided favorable feedback, often accompanied by suggestions for improvements, which opens up possibilities for the development of updated versions of existing games or the creation of new educational proposals.

<sup>2</sup><https://gamelab.wssu.edu/index.htm>

<sup>3</sup><https://github.com/rosedu/wouso>

<sup>4</sup><https://www.openttd.org/>

The mapping was guided by ten research questions designed to explore the current landscape of educational games for operating systems in depth. The questions focused on identifying: (Q1) which operating system concepts the games address; (Q2) whether the games were used with students and the number of students involved; (Q3) how the tests were conducted and their outcomes; and (Q4) the challenges encountered during game application. The study also investigated (Q5) how the games were incorporated into OS teaching contexts, (Q6) whether they were used to introduce new concepts or reinforce existing ones, (Q7) the genres of the games, (Q8) the platforms for which the games were developed, (Q9) the programming languages used, and (Q10) which educational games are currently available.

Together, the answers to these questions provided a comprehensive overview of the role and characteristics of educational games in the teaching of operating systems, highlighting trends, gaps, and opportunities for future development and research.

Furthermore, it was observed that simulators and educational games play an essential role in operating systems education. They combine theoretical concepts with interactive practices, aiding in content retention and skill development. The continuous study and improvement of these tools, along with the pursuit of new resources, are fundamental to ensuring comprehensive and solid training in operating systems, thereby contributing to the advancement of teaching and research in Computer Science. Areas such as threads, critical regions, mutual exclusion, and others have few examples of games, also presenting opportunities for studying less explored areas.

Although the games showed positive outcomes, further reflection is needed on how they affect deeper learning processes, such as conceptual understanding, long-term retention, or critical thinking—within the domain of operating systems. To address the reported challenges, such as limited documentation or technical barriers, it is recommended that future game-based initiatives be accompanied by educator guides, open-source code, and cross-platform compatibility. Besides, rather than isolated interventions, the integration of educational games into the operating systems curriculum should be planned throughout the course, aligned with specific learning goals and assessment strategies.

Future work aims to analyze these underexplored areas, aiming at the creation of both digital and non-digital educational games focused on teaching these concepts, thus filling the gap in the field of educational games for operating systems.

Games offer an opportunity for the development of reasoning, logic, and other essential skills in problem-solving, thus promoting academic learning and the decision-making necessary in the job market.

## Declarations

## Funding

This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES)

- Funding Code 001, the National Council for Scientific and Technological Development (CNPq) - CNPq Fellow - Brazil (311685/2017-0), and the Araucária Foundation (17.633.124-0).

## Authors' Contributions

VC is the main contributor to the project and responsible for writing the original draft. VC also contribute to the conceptualization. MCJ is in charge of project administration, conceptualization, investigation and contributes to writing, reviewing, and editing the document. FS and LA also contribute by reviewing and editing the document. FS also contribute to the investigation of the paper. All authors read and approved the final manuscript.

## Competing interests

The authors declare that they have no competing interests.

## References

- AbdelAziz, M. A., ElBakry, H. M., Riad, A. E.-D. M., and Senouy, M. B. (2020). The impact of using virtual reality on student's motivation for operating systems course learning. *Journal of E-Learning and Knowledge Society*, 16(2):25–33. DOI: <https://doi.org/10.20368/1971-8829/1135076>.
- Battistella, P. and von Wangenheim, C. G. (2016). Games for teaching computing in higher education—a systematic review. *IEEE Technology and Engineering Education*, 9(1):8–30.
- Cameron, M. (2023). *Parallel Islands: A Diversity Aware Tool For Parallel Computing Education*. PhD thesis, Virginia Tech. Tese de Doutorado.
- Christopher, W. A., Procter, S. J., and Anderson, T. E. (1993). The nachos instructional operating system. In *USENIX Winter*, pages 481–488.
- Clementino, E. G., da Silva, T. R., da Silva Aranha, E. H., and dos Santos, F. G. (2022). Jogos não digitais para ensino de computação—um mapeamento sistemático. In *Anais do XXXIII Simpósio Brasileiro de Informática na Educação*, pages 540–550. SBC. DOI: <https://doi.org/10.5753/sbie.2022.225240>.
- Costa, L. D. (2009). O que os jogos de entretenimento têm que os jogos educativos não têm. In *VIII Brazilian Symposium on Games and Digital Entertainment*, pages 8–10.
- Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2013). *Sistemas Distribuídos-: Conceitos e Projeto*. Bookman Editora.
- Cruz, E. H., Foleiss, J. H., Assunção, G. P., and Gonçalves, R. A. (2008). Ferramenta de simulação de processador para ensino de graduação e pesquisa científica. *Anais SUL-COMP*, 4.
- da Silva, F. F. and Aylon, L. B. R. (2022). Mannakdt: Uma abordagem prática para aprendizagem multimodal e multidimensional da educação 5.0. In *Anais Estendidos do XXVIII Simpósio Brasileiro de Sistemas Multímedia e Web*, pages 115–118. SBC. DOI: [https://doi.org/10.5753/webmedia\\_estendido.2022.227227](https://doi.org/10.5753/webmedia_estendido.2022.227227).



- de Barros Costa, E. and Rocha, H. J. B. (2018). Programação numa abordagem de aprendizagem baseada em resolução de problemas e jogos: um mapeamento sistemático. *SBC—Proceedings of SBGames*.
- de Camargo, V. H. S., Campano Junior, M. M., Silva, F. F., and Aylon, L. (2024). Mapeamento sistemático de jogos educativos voltados para o ensino de sistemas operacionais. In *Anais Estendidos do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 1200–1211, Manaus, AM, Brasil. SBC. DOI: <https://doi.org/10.5753/sbgames.2024.240929>.
- de Jesus Santos, A. P., da Conceição, D. P., das Virgens Santos, E., and de Araujo Cirqueira, L. (2020). Arena deadlock: Uso de atividades lúdicas na educação de nível superior. *Brazilian Journal of Development*, 6(3):14579–14589.
- DeLozier, C. and Shey, J. (2023). Using visual programming games to study novice programmers. *International Journal of Serious Games*, 10(2):115–136. DOI: <https://doi.org/10.17083/ijsg.v10i2.577>.
- Falkembach, G. A. M., Geller, M., and Silveira, S. R. (2006). Desenvolvimento de jogos educativos digitais utilizando a ferramenta de autoria multimídia: um estudo de caso com o toolbook instructor. *Revista Novas Tecnologias na Educação*, 4(1).
- Figueiredo, R. T., dos Santos, V. M. L., and Ramos, J. L. C. (2020). Speed schedule-jogo para auxílio no estudo das políticas de escalonamento em sistemas operacionais. *Informática na educação: teoria & prática*, 23(1 Jan/Abr).
- Freitas, G. M. B. d. (2023). Simulador de gerência de processos para sistemas operacionais. *Trabalho de Conclusão de Curso - Bacharel em Sistemas de Informação. Universidade Federal de Uberlândia - Faculdade de Computação*.
- Fukao, A. T., Colanzi, T. E., Martimiano, L. A., and Feltrim, V. D. (2023). Estudo sobre evasão nos cursos de computação da universidade estadual de maringá. In *Anais do III Simpósio Brasileiro de Educação em Computação*, pages 86–96. SBC. DOI: <https://doi.org/10.5753/educomp.2023.228209>.
- Gadelha, R. N., de Azevedo, R. R., de Oliveira, H. T., Neves, T. D., Souza, C. C., and da Silva, E. L. (2010). Os simulador: Um simulador de sistema de arquivos para apoiar o ensino/aprendizagem de sistemas operacionais. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1.
- Haddad, F., Filho, W. R., Ramos, V., Corrêa, C., and Peres, L. (2024). Mapeamento sistemático da literatura de jogos educacionais destinados ao ensino e aprendizagem de engenharia de software: uma análise do estado da arte. In *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 1256–1269, Porto Alegre, RS, Brasil. SBC. DOI: <https://doi.org/10.5753/sbgames.2024.241082>.
- Hilton, M. L. and Honey, M. A. (2011). *Learning science through computer games and simulations*. National Academies Press.
- Jones, D. and Newman, A. (2001). Rcos. java: A simulated operating system with animations. *Teaching package 1*.
- Julio, J., Campano Junior, M. M., Aylon, L., Fonseca, K., and Emmendorfer, L. (2024). Jogos educativos para estruturas de dados: Um mapeamento sistemático. In *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 1186–1199, Porto Alegre, RS, Brasil. SBC. DOI: <https://doi.org/10.5753/sbgames.2024.240933>.
- Kioki, E. Y., Santiago, P. P., and Soares, A. C. (2008). Um simulador didático como ferramenta de apoio ao ensino da disciplina de sistemas operacionais. *INICIA*, 37:40.
- Kitchenham, B. and Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075.
- Kurose, J. and Ross, K. (2010). Computer networks: A top down approach featuring the internet. *Peorsoim Addison Wesley*.
- Luccas, M. d. S. (2019). Jogos educacionais para ensino em sistemas operacionais. *Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação*.
- Machado, F. B. and Maia, L. P. (2000). *Fundamentos de sistemas operacionais*. Grupo Gen-LTC.
- Machado, F. B. and Maia, L. P. (2004). *Arquitetura de sistemas operacionais*, volume 4. LTC.
- Maia, L. P. (2001). Sosim: Simulador para o ensino de sistemas operacionais. *Dissertação de Mestrado - Universidade Federal do Rio de Janeiro*.
- Maziero, C. A. (2002). Reflexões sobre o ensino prático de sistemas operacionais. In *Anais do X Workshop sobre Educação em Computação (WEI2002)*.
- Medeiros, T. R., Souza, C. C., de Sousa, T. D., NS, R., Gadelha, E. L. d. S., and Júnior, J. B. D. (2011). Io simulator: Um simulador de dispositivos de entrada e saída para auxiliar o ensino de sistemas operacionais. In *Workshop de Educação e Informática. XXXI Congresso da Sociedade Brasileira de Computação*, pages 1647–1655. SBC.
- Moran, J. (2018). Metodologias ativas para uma aprendizagem mais profunda. *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática. Porto Alegre: Penso*, pages 02–25.
- Murphie, B. and Hansen, M. (2018). Teaching concurrency in a modern manner, flipped classroom or game-based learning. *Department of Computer Science and Media Technology. Malmö universitet/Teknik och samhälle*.
- Oliveira, R. A. and Souza, A. C. d. S. (2015). Swso-simulador web de sistemas operacionais. *Análise e Desenvolvimento de Sistemas. Instituto Federal da Bahia*.
- Pascotini, M. T. (2018). Proposta de um modelo para criação de jogos educativos. Master's thesis, Centro de Artes e Letras. Universidade Federal de Santa Maria (UFSM).
- Popović, M., Vladimir, K., and Šilić, M. (2018). Application of social game context to teaching mutual exclusion. *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, 59(2):208–219. DOI: <https://doi.org/10.1080/00051144.2018.1522462>.
- Quirino, T. M. F., Campos, C. C. V., and Oshima, R. M. S. (2017). O uso de jogos no ensino superior como estratégia pedagógica. In *Revista Tecnologias na Educação - Simpósio Nacional de Tecnologias Digitais na Educação (SNTDE)*, number 22.
- Reis, F. P. and Costa, H. (2009). Tbc-so/web: Software educativo para aprendizagem de polioticas de escalonamento de processos e de alocação de memória em sistemas operacionais. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE*,

- UFSC, Florianópolis.
- Rocha, A. R., Schineider, A., Alves, J. C., and de Abreu Silva, R. M. (2004). wxproc—um simulador de políticas de escalonamento multiplataforma. *INFOCOMP Journal of Computer Science*, 3(1):43–47.
- Rughiniş, R. (2013). Scaffolding a technical community of students through social gaming: Lessons from a serious game evaluation. *CSCL 2013 Proceedings. Volume 2: Short Papers, Panels, Posters, Demos, & Community Events*.
- Rutten, N., Van Joolingen, W. R., and Van Der Veen, J. T. (2012). The learning effects of computer simulations in science education. *Computers & education*, 58(1):136–153.
- Sanches, M. H. B. (2019). Jogos de entretenimento no ciclo educacional básico: critérios de aplicação e desenvolvimento de competências e habilidades. Master's thesis, Dissertação de mestrado (Tecnologias da inteligência e design digital). São Paulo.
- Santini, L., Junior, M. C., Felinto, A., and Aylon, L. (2022). Jogos no ensino de linguagens formais e autômatos: Um mapeamento sistemático. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 886–895, Porto Alegre, RS, Brasil. SBC. DOI: [https://doi.org/10.5753/sbgames\\_estendido.2022.226064](https://doi.org/10.5753/sbgames_estendido.2022.226064).
- Santini, L. F., Santini, A. L., Campano Junior, M. M., Track, M., Assumpção, M., and Aylon, L. (2023). Jogos educativos no ensino de circuitos digitais: Um mapeamento sistemático. In *Anais Estendidos do XXII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 814–825, Porto Alegre, RS, Brasil. SBC. DOI: [https://doi.org/10.5753/sbgames\\_estendido.2023.234063](https://doi.org/10.5753/sbgames_estendido.2023.234063).
- Sebesta, R. W. (2018). *Conceitos de Linguagens de Programação-II*. Bookman Editora.
- She, Y.-X., Lin, M.-H., Jong, B.-S., and Hsia, Y.-T. (2013). Using growing pet game in facebook to enhance students' learning motivation: In operating system course. In *2013 Learning and Teaching in Computing and Engineering*, pages 224–228. IEEE.
- Silva, E. O., Junior, W. M. V., and Carmona, J. V. C. (2021). Webjuvia: Simulador web de apoio ao ensino de gerência de memória na disciplina de sistemas operacionais. In *Anais do Simpósio Brasileiro de Educação em Computação*, pages 343–351. SBC. DOI: <https://doi.org/10.5753/educomp.2021.14502>.
- Souza, D. A. d. (2014). *Protótipo de um jogo educativo para o auxílio de ensino e aprendizagem em sistemas operacionais*. PhD thesis, Universidade Estadual do Piauí (UESPI)-Campus Professor Alexandre Alves de Oliveira-Parnaíba.
- Tanenbaum, A. S. and Bos, H. (2015). *Modern operating systems*. Pearson Education.
- Tanenbaum, A. S., Woodhull, A. S., et al. (1997). *Operating systems: design and implementation*, volume 68. Prentice Hall Englewood Cliffs.
- Unity (2024). Plataforma de desenvolvimento em tempo real do unity - 3d, 2d, engine vr e ar. <https://unity.com/pt> Acessado em fevereiro 2024.
- Weanquoi, P., Zhang, J., Yuan, X., Xu, J., and Jones, E. J. (2021). Learn access control concepts in a game. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE.
- Zhang, J., Yuan, X., Johnson, J., Xu, J., and Vanamala, M. (2020). Developing and assessing a web-based interactive visualization tool to teach buffer overflow concepts. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–7. IEEE. DOI: <https://doi.org/10.1109/FIE44824.2020.9274239>.
- Zhu, J., Alderfer, K., Furqan, A., Nebolsky, J., Char, B., Smith, B., Villareale, J., and Ontañón, S. (2019). Programming in game space: how to represent parallel programming concepts in an educational game. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, pages 1–10. DOI: <https://doi.org/10.1145/3337722.3337749>.