

CSCape: A Video-based Escape Room Framework (not Only) for Computer Science

Johannes Schildgen

Regensburg University of Applied Sciences, Germany

johannes.schildgen@oth-regensburg.de

Abstract: CSCape is a software framework for interactive escape-room adventures in computer science and other disciplines. It runs on a lightweight computer, such as a Raspberry Pi, which is connected to a projector or a TV. Initially, it shows a black screen. However, when a task is completed, a video plays, and the story continues. Educators must provide video files or texts and images, and develop Python functions to verify whether a task is solved or not. Potential examples in computer science include: Is the correct data in a database or a Google spreadsheet? Is the network correctly configured? Is an application running on a specific server? Is a file or folder accessible over the network, and does it contain the correct data? Additional examples: Students fill out and submit a Google Form. This can be a multiple-choice quiz or a complex text input. The submitted data is checked with Python. CSCape can be combined with real-world puzzles and items within the room. For instance, a character in the video might say that you need to solve a mathematical calculation. The solution is a 3-digit code that opens a combination lock. Behind that lock is a note on a paper and a USB stick containing data that needs to be imported into a database. After that, the next video is shown. Players solve tasks either on their own notebook or phone, or on pre-configured computers, all without supervision. The software sends push messages to the educator's phone to inform them about the team's current game progress. Furthermore, Wi-Fi smart plugs can be integrated to control lights and other devices, allowing them to be powered on and off when a task is completed. CSCape is designed to achieve a wide range of learning objectives, including technical skills, creativity, problem-solving, teamwork, and the ability to work under time pressure. Through storytelling, interactive challenges, and real-world puzzles, it sparks curiosity and sustains motivation, making learning not just effective but fun. The framework is open-source and easily customizable. Escape-room games can be played by people of all ages, from children to adults, and are adaptable for use in computer science and other academic disciplines.

Keywords: Interactive learning, Game-based learning, Escape-room software framework

1. Introduction

Escape rooms are team-based games in which players solve puzzles and complete tasks within one or more rooms to accomplish a goal. (Nicholson 2015) The first live-action escape rooms originated in Japan in 2007; today, they are played all over the world. Typical escape-room stories are inspired by role-playing games and feature themes such as prison breaks and murder mysteries. Players often take on specific roles within the story—such as detectives, prisoners, or explorers—and many scenarios are set in the past, creating an immersive, historical experience. In recent years, escape rooms have also been adapted for educational purposes, with game-based elements used to enhance learning, foster collaboration, and increase participants' motivation. (Sanchez and Plumettaz-Sieber 2018; Fotaris and Mastoras 2019) There are both physical and digital escape rooms, allowing participants to either interact with real-world objects in a physical environment or solve puzzles virtually via online platforms. (Makri, Vlachopoulos and Martina 2021)

In this paper, we present a software framework for creating video-based physical escape rooms that enables the design and execution of interactive puzzles. The name CSCape stands for Computer Science Escape Rooms; however, as we will show later in this paper, the framework is not limited to computer science education and can be applied to other disciplines as well. The framework is open source and runs on low-cost devices like Raspberry Pis. Educators can create their own escape rooms by developing custom Python functions for game logic and providing story material, such as videos. These components can be shared and reused by other educators, fostering a marketplace-like ecosystem for exchanging escape room modules and learning scenarios.

2. Related Work

Escape rooms have increasingly been studied and adopted as an educational tool in recent years. (Taraldsen et al. 2022) Prior work has shown that game-based learning environments can improve student motivation, engagement, and collaborative problem-solving skills. (Borrego et al. 2017) In particular, escape-room-like activities have been applied in computer science education to teach topics such as programming, algorithms, and databases in an engaging and interactive manner. (Lathwesen and Belova 2021; Hacke 2019; López-Pernas et al. 2019)

Several digital escape room platforms exist that allow the creation of virtual puzzle environments. These systems typically focus on fully online experiences and often provide predefined templates or only limited customization options. In contrast, physical escape rooms integrate hardware components such as sensors, actuators, or external devices to create more tangible and immersive interactions.

Other approaches explore using Internet of Things (IoT) devices and microcontrollers to connect physical artifacts to digital logic. However, these systems are often tightly coupled to specific hardware setups and lack a flexible, reusable abstraction layer for educators.

3. The CSCape Framework

In a typical escape-room setup, CSCape runs on a device such as a Raspberry Pi connected to a projector or TV. On this device, two main processes run: a web browser displaying a full-screen presentation with black slides and embedded videos, and the runtime system that controls the game logic and interactions. The players interact with the system using separate devices such as PCs, tablets, smartphones, or laptops. These devices are connected via a local network or the internet, enabling the system to verify whether and when a task has been solved.

Figure 1 shows the components of the CSCape framework and how they interact with each other:

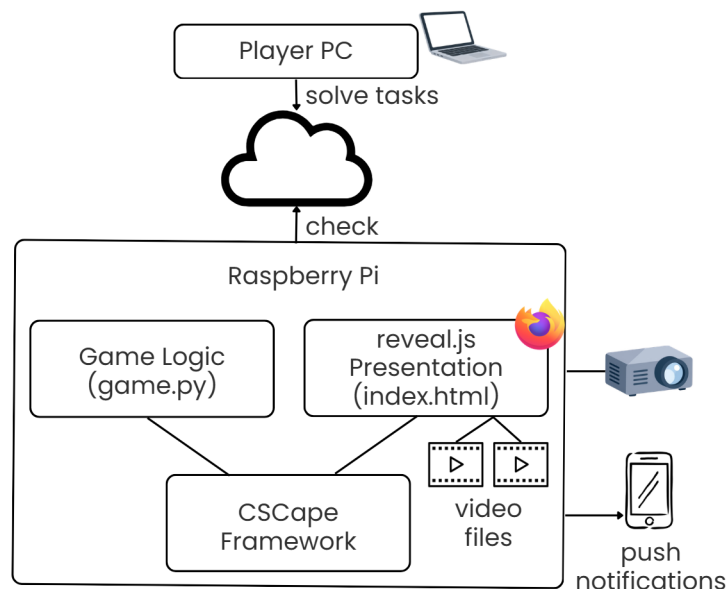


Figure 1: The components of the CSCape framework and how they interact with each other

- **Player devices:** Players use one or more PCs—either pre-configured devices provided in the room or their own laptops—to interact with the game.
- **game.py:** Custom check functions that verify whether players have successfully solved a task.
- **cscape.py:** The framework orchestrates the game logic, delegates task verification to game.py, and sends push notifications to the educator’s device to update them on the players’ progress.
- **index.html:** Full-screen presentation slides, indicating which task players must solve next and specifying which video to play once the corresponding check function in game.py confirms the task is completed.

To host a ready-made escape room, the required files need to be stored on a Raspberry Pi. Then, the provided run.sh script will automatically start the Python server and open the browser. This makes the system ready for players to begin immediately.

3.1 Creating an Escape Room

To create a new escape-room scenario, the story videos must be placed in the videos folder. The corresponding check functions need to be implemented in the game.py script to verify task completion. Finally, the index.html file must be updated to define the sequence of tasks and specify which video to play after each task is completed. This prepares the custom scenario for use with the CSCape framework.

The game.py script includes an `__init__` method for initialization and any number of `check_*` methods to verify task completion:

```
class Game:
    title = "My Escape Room"

    def __init__(self, game_data_store):
        # Prepare the environment when the server starts
        # Create tables, clean up remnants from previous game sessions, ...

    def check_firsttask(self):
        # Return True when the level is solved
        expected_solution = 5
        return solution == expected_solution
```

Each task in the escape room corresponds to a dedicated `check_*` function in the game.py script and one slide in the index.html:

```
<section data-cscape-check="check_firsttask"
        data-background-video="videos/hello.mp4"
        data-background-size="contain"
        data-autoplay></section>
```

The system continuously polls the `check_firsttask` method every five seconds. As soon as the method returns True, the video hello.mp4 is played. The system then waits for the next task to be completed, repeating the process for each subsequent task.

Basic Python skills are required to implement and customize the game.py script. However, AI tools like ChatGPT, Gemini, Claude Code, or Mistral Vibe can provide valuable assistance—whether for writing code, debugging, or generating creative solutions—making the process more accessible for educators and developers.

For reusing check methods for multiple tasks, it is possible to add an additional parameter: `def check_something(self, param)`. This parameter can then be set in the HTML code using the `data-cscape-check-param` attribute.

More details on how to create an escape-room scenario and how to check whether tasks are solved can be found in the example in Section 5 of this paper.

3.2 Push Messages

The CSCape framework can send push notifications to the educator’s mobile phone via Telegram. To enable this, a Telegram bot must be created using BotFather. After that, the Telegram API token and the chat ID must be stored within the config.ini file:

```
[telegram]
telegram_push = true
token = YOUR_BOT_TOKEN
chat_id = YOUR_CHAT_ID
```

Notifications are sent whenever a task is completed. Each notification includes the escape room title, the task name, and the current progress (e.g., “My Escape Room - Level 1 solved: check_firsttask”).

When hosting more than one escape room simultaneously, each must use a unique title (e.g., “My Escape Room #1” and “My Escape Room #2”). Furthermore, it needs to be ensured that the system can differentiate between rooms—either by using unique usernames, separate data files, or room-specific task variants.

3.3 Parallel Tasks

By default, the game follows a linear story progression, where tasks are checked and solved one after another. However, in CSCape, it is also possible to allow tasks to be completed in any order after a certain point. To implement this, nested subsections within a slide group in index.html can be used. A subsection is a

<section> element within another <section> element. This allows multiple video files or other types of feedback to be provided for several sub-tasks, which are checked simultaneously. When a sub-task is completed, the corresponding video is played. Once all sub-tasks within the group are solved, the system advances to the next main slide.

In an example project, players find a set of multiple task cards. A character within a video says, „But you’ll have to split up now, or you won’t make it in time.“ This feature enables player groups to work on separate tasks in parallel, enhancing collaboration and gameplay flexibility.

3.4 Actions

As soon as a task is completed, the presentation moves to the next slide. This slide can contain a video, an image, or text. In addition, CSCape allows the development of custom *action* methods that are linked to a specific check method and executed as soon as the task is completed. This allows for the execution of additional user-defined code.

Example actions include creating a new file or adding new data to a database, which can serve as the basis for the next task. Other options include sending messages via SMS, Telegram, or Slack to players, interacting with a smart speaker or a remote display, printing a document on a connected printer, or integrating IoT devices. Smart plugs, Zigbee light bulbs, or other devices can be turned on within action methods to create immersive environmental changes, signal task completion, or unlock new physical interactions—such as illuminating a hidden clue, powering on a device needed for the next challenge, or triggering a visual or auditory cue that enhances the storytelling and engagement of the escape room experience.

When creating an action method, this method can be referenced within the HTML code:

```
def example_action(self):
    turn_on_light()

<section data-cscape-check="check_example" data-cscape-action="example_action" .../>
```

In this example, the light turns on as soon as `check_example` indicates that the associated task has been completed.

3.5 Accessing User Input

As shown in Figure 1, CSCape runs on a lightweight device such as a Raspberry Pi connected to a projector or TV. Players interact with the system using separate devices—such as PCs, tablets, smartphones, or laptops—where they complete tasks and submit their solutions. There are three ways the CSCape device and the players’ devices are connected:

- **Shared Resource:** The player devices and the CSCape device access the same shared database or file on a remote server, in the cloud, or on the Raspberry Pi itself.
- **Direct Player-Device Access:** This approach is generally not feasible when players use their own devices, as it requires pre-configured player devices with properly set up networks and explicit access permissions.
- **Game Data Store:** The player devices send input data to CSCape’s Game Data Store endpoint.

3.6 Game Data Store

To bridge the gap between player devices and the central CSCape system, the Game Data Store endpoint serves as a dedicated interface for receiving user input from external programs. Whether players enter a code, submit a file, or trigger an event, their actions can be sent directly to that endpoint, enabling the framework to store input data in a key-value store for the current game session. Within check methods, this input data can be accessed from the store and validated. This seamless integration ensures near-real-time interaction without the need for a central external data store.

```
def check_firsttask(self):
    return self.game_data_store.get("code") == "1234"
```

Additionally, the Game Data Store can be used to persist session-specific information—such as player names—that can be displayed in the frontend presentation. For example, if players are tasked with entering their team name into a database table, the corresponding check or action method can store this value in the Game Data Store:

```
self.game_data_store.store("teamname", teamname)
```

Within the slides (index.html), this value can then be dynamically received. This allows the team name to appear, for instance, in a speech bubble on the slide, providing immediate visual feedback:

```
<div class="speech-bubble">
  Hello <span data-cscape-get="teamname"></span>!
</div>
```

By storing these values in the key-value store, the presentation layer can render them in real time, ensuring that players see personalized or context-relevant information as they progress through the game. This not only enhances interactivity but also makes the experience more engaging for players, as they can see their direct influence on the game's presentation. It's important to note that such dynamic updates are not possible in pre-produced videos or audio messages but are mainly for text content or text-to-speech audio on the slides.

4. Escape Room Examples

The vision of CSCape is to create a comprehensive marketplace where educators can easily publish, access, adapt, or directly reuse pre-designed escape room scenarios. While CSCape is primarily designed for computer science education, its flexibility makes it equally powerful for a wide range of disciplines. Below are some creative escape room ideas that demonstrate the versatility of the CSCape framework:

Working with databases: (Barra et al. 2023; Malik et al. 2024) At the start of the game, players find database credentials. As soon as they are connected, the story continues, and tasks have to be solved directly in the database using the SQL query language. CSCape checks whether the correct data is inserted into a table with simple SELECT queries on those tables. Many databases also provide an audit log feature. MySQL calls this the *general query log*. When this feature is enabled, all SQL statements from all clients are logged into a table. CSCape can access this table, re-execute the same SELECT, and check whether it returns the desired output. For teaching advanced and new SQL skills, hints can be hidden on paper within the room, with video clues guiding players to their locations. Additionally, a USB stick containing datasets can be hidden in the room. Players must find the USB stick and import the data into the database.

Linux console skills: Players receive SSH access credentials to connect to a Linux server and must solve a series of tasks directly on the console. Challenges include creating and navigating directories, editing configuration files, setting file permissions, and troubleshooting system errors. Hidden clues—such as notes taped to the monitor or messages in log files—guide players toward the correct commands and solutions.

Cybersecurity: (Beguín et al. 2019; Seebauer, Jahn and Mottok 2019) This scenario builds on the previous example by adding a cybersecurity layer. Players must secure a compromised system, identify vulnerabilities, and patch security flaws—all while working directly on the Linux console. Additional challenges include decrypting encoded messages, analyzing network traffic, and preventing simulated data breaches.

Programming skills: (López-Pernas et al. 2019; Humble and Mozeliuss 2022) Players are tasked with solving a series of programming challenges in a realistic context. They might need to debug existing code, write new scripts and functions, or develop small applications to solve specific problems. This scenario is more complex to implement compared to others, as it requires retrieving and executing player-written code. Unlike the continuous checks in other scenarios, players must explicitly submit their code—either by pushing it to a Git repository or using dedicated software that is currently under development. The software will allow players to develop and run their programs (e.g., in Python) on their local machines. Once they believe they have solved the task, they submit their code via the Input Receiver. On the CSCape device, the submitted code is executed and validated using predefined tests to ensure it meets the requirements.

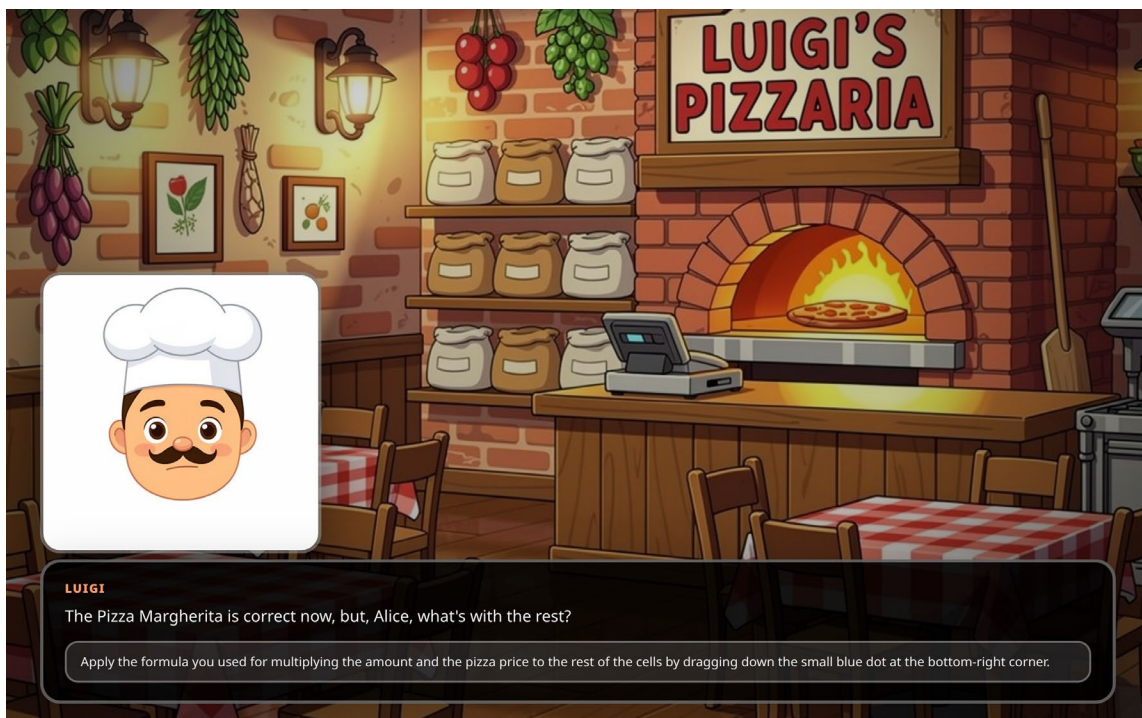
Working on a spreadsheet: (Perry 2026; Drier 2001) Players have access to an Excel document, such as via Google Sheets or Microsoft OneDrive, where they must complete a variety of tasks. This setup allows for a wide range of activities. The most obvious one is applying and understanding Excel formulas to solve problems or analyze data. Furthermore, the spreadsheet can be used to let players enter the results of mathematical calculations, which are then verified by CSCape. In another scenario, players record their measurement data from experiments (e.g., chemistry, physics, or biology) into the spreadsheet. If the values are plausible, the story continues.

5. Case Study: Luigi's Missing Pizza Order

We now present a simple example escape room where players work on a Google Spreadsheet. For this, educators need to create a new spreadsheet and copy its public URL into the CSCape configuration file. Players can open this URL on their own computer or on their smartphone to view and edit the spreadsheet. By the end of this game, learners will be able to use basic Excel formulas for calculations.

D4			$\text{fx} = \text{B4} * \text{C4}$	
	A	B	C	D
1	Luigi's Pizza	New Order	Your Name:	Alice
2				
3	Pizza	Quantity	Price	Total
4	Margherita	4	10.00 €	40.00 €
5	Pepperoni	1	11.50 €	
6	Diavola	2	12.50 €	
7	Quattro Formaggi	1	12.50 €	
8	Calzone	1	15.80 €	
9				

Gameplay: Players enter the room and find a story handout that also contains a shortened spreadsheet link and a QR code. The story is about Luigi's Pizzeria. When Luigi checks his delivery list for a big order, it's completely blank. Now he needs help. The first task is that the players need to open the spreadsheet and enter their name in cell D1. This way, they learn how to work with spreadsheets in general. As soon as this first task is solved, a sad Luigi appears on the screen. He is desperately saying: "I last saw the order form when I was checking the street map for the fastest delivery routes.". Players then find the map in the room, open it, and find a hidden pizza order sheet. The player input the pizza quantities into column B, then the story continues. Now, Luigi is happy: "Ah, perfetto! The order is back. Can you help me to calculate the total for all Pizza Margherita?". The players have to write a multiplication formula into cell D4 to complete the task. The next task is to apply this to all items, and the final task is to use the SUM function to calculate the total amount for the whole order. When all tasks are solved, Luigi congratulates the participants.



Technical implementation: The game.py contains one constructor method (`__init__`) where the spreadsheet URL is loaded from the configuration, two check methods, and two helper methods `get_sheet_data` and `cell_value` which extracts values from specific cells.

- `__init__`: We will load the Google Spreadsheet data in CSV format using Python's `requests` library. There are also special Google SDKs for working with spreadsheets, but for this simple use case, it's sufficient to simply retrieve the spreadsheet data in CSV format. Mind that in this format, all formatting (styles, colors) and formulas are discarded. For example, when the user writes `=B4*C4` into a cell, the CSV export shows the number 40 in that cell. On the one hand, this makes it more flexible because often, more than one solution is correct. On the other hand, this allows players to bypass the intended logic by hardcoding results, and write 40 directly into the cell instead of the formula that calculates this value.
- `get_sheet_data`: This helper method loads the data from the spreadsheet into a Python list of lists. This method will be called every five seconds to refresh the changes made by the players.
- `cell_value(cell)`: This method returns the value that's in a specified cell. For example, `cell_value("A1")` returns the value in the top-left cell.
- `check_cell_not_empty(param)`: With this method, we can check whether one or more cells contain values. We will use this to check whether players have written their name into cell D1 in the spreadsheet. If so, the method returns True and the task is solved. Additionally to this behavior, the method stores the values from these cells into the game data store (see Section 3.6) so that we can retrieve the player's name in the story's HTML file.
- `check_cell_value(param)`: This method takes a predicate string as a parameter, for example `D4=40`. Multiple predicates can be combined with the `&` operator: `B4=4&B5=1&...`. This way, we check whether the players have correctly inserted the numbers from the order form they found. The function checks whether all specified cells match the expected values. When this is the case, the function returns True, otherwise it returns False. This method is also used for all other tasks. Players need to write the formula `=B4*C4` to calculate the total price for all Pizza Margherita. As the price for Pizza Margherita (C4) is 10 EUR, and the quantity (B4) is four, the function returns True if the value in the result cell D4 is 40. Next, the players now need to apply the formula from the previous task to a few next rows. We check whether all multiplication results are correct within the cells D4:D8. To calculate the order total, players should use the SUM function. The method returns True if the cell D9 contains the correct number.

Instead of playing pre-recorded video clips, the Luigi's Pizza escape room uses the CSCape Story Styles plugin which is part of the CSCape framework. It shows a customizable background image, an avatar image representing the character, and the dialogue text. The Web Speech API which is supported by all modern browsers is used to synthesize speech from text. This makes it also possible to speak the player's name (here: Alice) that the players have written into the spreadsheet.

During the story, players learn how to use Excel formulas, multiplication, how to replicate formulas across multiple cells, and how to use the SUM function. Nevertheless, it is possible to bypass the formulas by entering hardcoded results directly into cells. We simply accept this as a valid solution.

The spreadsheet-based escape room is highly customizable. It can be easily adjusted to other themes and learning topics as well. This can be done without modifying any Python code. Educators simply have to edit the dialogue texts in the HTML file, replace image files, and modify the task predicates that need to be fulfilled for each task, e.g. `D4=40`.

6. Conclusion

In this paper, we introduced CSCape, an open-source software framework that enables educators to design and deploy interactive, video-based escape-room learning experiences. By integrating storytelling, real-world puzzles, and customizable digital tasks, CSCape transforms traditional education into a dynamic hands-on adventure. The framework supports a broad spectrum of learning objectives—from technical skills such as programming and database management to interdisciplinary challenges. Its modular architecture allows educators to create customized scenarios that align with their curriculum, fostering creativity, problem-solving, and teamwork in an engaging environment.

While CSCape currently empowers educators to develop and customize their own escape-room scenarios, our vision includes creating a community-driven marketplace. This marketplace would enable educators to share, adapt, and reuse scenarios, further enriching the diversity and accessibility of learning experiences. By combining physical and digital interactions, CSCape not only enhances student motivation but also prepares learners to tackle real-world challenges effectively under time pressure.

Acknowledgements:

Special thanks to Marlene Stegmann and Elias Wagner for developing and testing example escape rooms using the CScape framework, and to Marlene Stegmann for the Story Styles Plugin.

Source code: CScape is open source and available at <https://github.com/jschildgen/cscape>. The repository also includes the Story Styles plugin and the Spreadsheet Escape Room “Luigi’s Missing Pizza Order”.

Ethics Declaration: This study did not involve any ethical considerations that needed clearance.

AI Declaration: AI-based tools were used for minor language refinement, and all generated suggestions were reviewed and edited by the author.

References

- Barra, E. et al. (2023) ‘Empowering database learning through remote educational escape rooms’, *IEEE Internet Computing*, 28(1), pp. 18–25.
- Beguín, E. et al. (2019) ‘Computer-security-oriented escape room’, *IEEE Security & Privacy*, 17(4), pp. 78–83.
- Borrego, C. et al. (2017) ‘Room escape at class: escape games activities to facilitate the motivation and learning in computer science’, *JOTSE*, 7(2), pp. 162–171.
- Drier, H.S. (2001) ‘Teaching and learning mathematics with interactive spreadsheets’, *School Science and Mathematics*, 101(4), pp. 170–179.
- Fotaris, P. and Mastoras, T. (2019) ‘Escape rooms for learning: a systematic review’, in *Proceedings of the European Conference on Games Based Learning*, pp. 235–243.
- Hacke, A. (2019) ‘Computer science problem solving in the escape game “Room-X”’, in *International conference on informatics in schools: situation, evolution, and perspectives*. Cham: Springer International Publishing, pp. 281–292.
- Humble, N. and Mozelius, P. (2022) ‘Refurbishing the educational escape room for programming: lowering the threshold and raising the ceiling’, in *16th European Conference on Game Based Learning (ECGBL 2022)*.
- Lathwesen, C. and Belova, N. (2021) ‘Escape rooms in STEM teaching and learning—prospective field or declining trend? A literature review’, *Education Sciences*, 11(6), p. 308.
- López-Pernas, S. et al. (2019) ‘Analyzing learning effectiveness and students’ perceptions of an educational escape room in a programming course in higher education’, *IEEE Access*, 7, pp. 184221–184234.
- Malik, S.I. et al. (2024) ‘Introducing an online escape room to enhance students’ database concepts’, in *Practices and implementation of gamification in higher education*. Hershey, PA: IGI Global Scientific Publishing, pp. 215–231.
- Makri, A., Vlachopoulos, D. and Martina, R.A. (2021) ‘Digital escape rooms as innovative pedagogical tools in education: a systematic literature review’, *Sustainability*, 13(8), p. 4587.
- Nicholson, S. (2015) Peeking behind the locked door: A survey of escape room facilities.
- Perry, S.C. (2026) ‘Supporting first-year chemistry undergraduates in data analysis through a gamified Excel escape room’, *Journal of Chemical Education*.
- Sanchez, E. and Plumettaz-Sieber, M. (2018) ‘Teaching and learning with escape games: from debriefing to institutionalization of knowledge’, in *International conference on games and learning alliance*. Cham: Springer International Publishing, pp. 242–253.
- Seebauer, S., Jahn, S. and Mottok, J. (2019) ‘Escape rooms for teaching IT-security’, in *ICERI2019 Proceedings*. IATED, pp. 6815–6824.
- Taraldsen, L.H. et al. (2022) ‘A review on use of escape rooms in education—touching the void’, *Education Inquiry*, 13(2), pp. 169–184.