



UNIVERSITY OF PIRAEUS

School of Information and Communication Technologies

Department of Informatics

Thesis

Thesis Title:	Education Through Video Games
Τίτλος Διατριβής:	Η Εκπαίδευση μέσα από τα ηλεκτρονικά παιχνίδια
Student's name	Tsamis Georgios
Father's name	Ioannis
Student ID Number:	Π21178
Supervisor	Panagiotis Tsakonas, Professor

June 2026/Ιούνιος 2026

Copyright ©

Copying, storing, and distributing this work, in whole or in part, for commercial purposes is prohibited. Reproduction, storage, and distribution for non-profit, education, or research only is allowed, provided the source is mentioned, and this message is preserved. The content of the document reflects the author's views and conclusions and does not necessarily represent the official views and conclusions of the University of Piraeus. As the author of this work, I declare that this work is not a product of plagiarism and does not contain material from uncited sources.

ACKNOWLEDGEMENTS

I would like to express my special thanks to my family for their patience, encouragement and constant support throughout my studies. I would also like to showcase my gratitude to my supervisor for their supervision, their constructive comments, and their continuous support in the preparation of this thesis. Their knowledge and experience in software engineering and educational technology were a great source of guidance and direction throughout all of this. I would also like to thank the professors of the department of Informatics at the University of Piraeus for contributing towards my education on the subjects of software development, algorithms and web technologies.

AI-based assistance was applied to check grammar and improve text in this project. However, all design decisions and coding were performed manually by the author without using any generative AI tools.

ABSTRACT

This paper is a documentation of the design and implementation of an educational dungeon game created in HTML, CSS, JavaScript, and Three.js. The player is a character in a succession of three-dimensional rooms where one is required to fight enemies by answering questions and finding hidden chests with hints of knowledge. This way, they are expected to answer informatics questions to unlock exit doors. Errors send the player back to 'penalty paths' ensuring the importance of correct recall. Progressive learning content is gathered in an in-game journal, which aids in spaced repetition and memory retention. The design of the system, game mechanics, educational design rationale, and implementation details are fully discussed. The project is a proof-of-concept in the use of a browser-based 3D game environment as an excellent tool to teach computer science topics at the undergraduate level.

Table of Contents

ACKNOWLEDGEMENTS	3
ABSTRACT.....	4
1. INTRODUCTION.....	8
1.1 Background and Motivation	8
1.2 Problem Statement	9
1.3 Research Objectives	10
1.4 Expected Contribution.....	11
1.5 Thesis Structure.....	11
2. LITERATURE REVIEW AND THEORETICAL BACKGROUND.....	13
2.1 Educational Games: Definition and History	13
2.2 Game-Based Learning Frameworks	14
2.3 Dungeon-Style Games as a Genre.....	17
2.4 Three.js and Browser-Based 3D Game Development.....	18
2.5 Related Works.....	20
2.6 Memory, Recall, and Learning Through Gameplay.....	21
2.7 Summary and Research Gap	23
3. SYSTEM DESIGN AND ARCHITECTURE	24
3.1 Game Concept and Educational Philosophy.....	24
3.2 Game Mechanics Overview	25
3.2.1 Room Exploration and Navigation.....	25
3.2.2 Question and Answer Door-Unlock System.....	26
3.2.3 Wrong-Answer Penalty Paths	27
3.2.4 Enemy Encounters and Scroll-Drop System.....	27
3.2.5 Hidden Chests & Wall Inscriptions.....	28
3.2.6 Player Journal System	29
3.3 System Architecture	30
3.4 Data Flow and State Management.....	31
3.5 Level and Room Design Logic	32
4. IMPLEMENTATION.....	35
4.1 Development Environment and Tools	35
4.2 Three.js Scene Setup: Renderer, Camera, and Lighting.....	36
4.3 Character Controller and Movement	37
4.4 Room Generation and Dungeon Layout	40
4.5 Question Engine: Loading, Evaluation, and Feedback	41

4.6 Enemy System and Scroll-Drop Mechanism	43
4.7 Chest Interaction and Wall-Text Rendering	44
4.8 Journal UI: Collection and Display	45
4.9 Game State Management	46
4.10 UI/UX Design: Menus, HUD, and Overlays	47
4.11 Challenges Encountered and Solutions	49
5. TESTING AND EVALUATION.....	52
5.1 Testing Methodology.....	52
5.2 Functional Testing.....	53
5.3 Educational Effectiveness Assessment.....	55
5.4 User Testing	57
5.5 Performance Testing.....	58
5.6 Discussion of Results.....	59
6. CONCLUSION	60
6.1 Degree to Which Objectives Were Achieved	61
6.2 Limitations	61
6.3 Innovative Elements and Contribution	62
6.4 Future Work Directions	63
7. BIBLIOGRAPHY	65
Table of Abbreviations and Acronyms.....	77

Table of Figures

Figure 1: Game-based Learning framework	15
Figure 2: Bloom's taxonomy alignment across game levels (Vanderbilt University Center for Teaching, n.d., via Management and Strategy Institute, 2023).....	18
Figure 3: Ebbinghaus forgetting curve and game-based countermeasures	21
Figure 4: Main Game loop.....	40
Figure 5: Level 1 question followed by a wall inscription.....	42
Figure 6: Question engine flow	43
Figure 7: Proximity interaction system	46
Figure 8: Room UI interface seen in level 1 of the game	48
Figure 9: Three-tier testing strategy applied in this evaluation	53
Figure 10: Functional test pass rates by mechanic category	54
Figure 11: Average frame rate by browser and device profile during Level 3 play-through	58
Figure 12: Future development roadmap across four phases	63
Figure 13: Room UI near a chest	69
Figure 14: Settings UI.....	70
Figure 15: Opened chest.....	70
Figure 16: Combat Question and UI before the fight	71
Figure 17: Correct answer UI from door question	72
Figure 18: Closed door before answering the question	73
Figure 19: Opened door after the correct answer	74
Figure 20: Final portal at the end of level 1	75
Figure 21: Inscription at the beginning of level 2	76

Table of Tables

Table 1: System architecture layers and their components	30
Table 2: Principal game events and their state and UI consequences	32
Table 3: Level composition summary (Source: author)	33
Table 4: Development tools and technologies (Source: author)	36
Table 5: Core Three.js scene configuration parameters (Source: author)	36
Table 6: Room configuration parameters (Source: author)	41
Table 7: Educational design criteria assessment against Stasiak (2023) framework (Source: author).....	56
Table 8: Research objective achievement summary	61

1. INTRODUCTION

1.1 Background and Motivation

The connection between playing and learning dates back to ancient times (Riede et al., 2018). However, systematic investigation into the use of games as means of teaching has emerged quite recently in the academic environment (Alotaibi, 2024). Notably, the early learning environment of computerized education including LOGO (Papert, 1980), revealed that playful interaction is an effective means through which learning takes place; this laid down the foundation of Game-Based Learning. In the last two decades, there has been an increase in the application of technology and the emergence of digital platforms, which have revolutionized the world of education and provided new ways of engaging that cannot be achieved through the use of conventional teaching methods. The evolution of Game-Based Learning (GBL) as a separate academic field is an indication of the realization that interactive and purposeful digital platforms can promote deep learning, continuous engagement, and even the growth of higher-order thinking abilities (Camacho-Sánchez et al., 2023; Mayer, 2019). In the sphere of computer science education, the issue of student engagement becomes especially relevant. Researchers have consistently found that students of technical subjects experience difficulties in comprehending abstract notions and lose motivation because of this fact within lecture-based classes, as well as find no correlation between theoretical knowledge and practice (Dicheva et al., 2015; Swacha, 2021).

As is known, video games consist of rules, tasks, feedback, and rewards, all of which correlate strongly with learning theories, such as Vygotsky's Zone of Proximal Development (ZPD) and Csikszentmihalyi's theory of Flow (Csikszentmihalyi, 1990; Alotaibi, 2024). Specifically, dungeons, or games of exploration in an environment composed of

interconnected rooms and puzzles with the gradual increase in challenge levels and resource collection, can provide an excellent structural basis for learning activities. Systematic exploration and completion of puzzles or combats associated with knowledge form a good correlation between gameplay and learning.

As a result of advances in web technologies, the creation of immersive three-dimensional virtual environments has become a feasible task through libraries such as Three.js, based on WebGL technology (Dirksen, 2023). Such an approach is highly advantageous when implemented in an educational setting due to the lack of restrictions in terms of software installations required for use. The current research stems from the alignment of multiple considerations. These are: the promising possibilities offered by GBL, the requirements of Computer Science education, the inherent structures of the dungeon games genre, and finally, the availability of 3D rendering technologies on the Web platform.

1.2 Problem Statement

Even though the amount of literature discussing the use of educational games is increasing, the application of educational games that promote efficient learning processes in Computer Science courses is still not widespread. There are several reasons that may account for the above issue. Firstly, commercial games rarely correlate with academic learning goals. Secondly, educational games that were specifically created for learning purposes require more financial resources, professional skills in game designing, and pedagogical practices (Prensky, 2007; Dondlinger, 2007). Thirdly, many game designs proposed by researchers are based on two-dimensional interactive worlds with little to no immersion potential (Lukka & Palva, 2023).

Another aspect that needs to be discussed relates to the inclusion of assessments in game designs. Unlike traditional assessment procedures conducted in a regular learning

environment, which are associated with students' nervousness and fear, the use of educational games makes it possible to integrate assessment into the narrative, where failure will lead to the choice of the wrong path but will not influence the final grades of learners negatively (Shute, 2011; Rajcsanyi-Molnar et al., 2025). The particular problem being studied in this thesis can thus be stated in the following manner: how can the conventions associated with the dungeon game genre be utilized in order to provide formative assessments and progressive learning within a three-dimensional browser environment applicable to Computer Science education at an undergraduate level?

1.3 Research Objectives

The main goal of this thesis is to create a complete, playable educational dungeon game full of educational content about informatics, which will be developed using HTML, CSS, JavaScript, and Three.js frameworks. The following subsidiary objectives are followed in this work:

- To perform a systematic review of the relevant literature regarding game-based learning, learning game design, and 3D development using a browser.
- To create a game architecture that incorporates formative assessment by way of question and answer at gameboard pivots (door unlocking, enemy combat).
- To build a knowledge accumulation system (drops, chests, wall inscriptions) that enables spaced exposure and progressive recall.
- To create a player log which collates knowledge items gathered, as a reference and revision aid in-game.
- To test the implemented system using a structured approach to test the function correctness.

- To critically consider the educational potential of the approach and directions for future development.

1.4 Expected Contribution

The thesis contributes to the field of educational technology and web-based game development in a number of ways. Technically, it shows a full implementation of a multi-level, 3D game, given purely in the browser, with open web standards and the Three.js library. The architecture here can be replicated by teachers and developers wishing to develop similar systems without relying on commercial game engines.

On the pedagogical level, the project embodies a particular approach to embedded formative assessment, where incorrect answers have realistic consequences in the game, wrong-path rooms are designed to operationalize the cost of lack of knowledge, and a continuous journal provides for ongoing consolidation of knowledge gained. This model is an extension of the current GBL framework by incorporating the features of memory and recall by means of scrolling, chests and wall inscriptions, which spread educational material throughout the game's environment, not just at specific assessment locations. Finally, the project is part of a larger body of work that demonstrates the viability of 3D technologies delivered through the web for pedagogically complex game environments accessible to undergraduate students with no particular hardware or software requirements.

1.5 Thesis Structure

The remainder of this thesis is organized as follows: The literature review in Chapter 2 includes a review of the literature related to the game-based learning theory, the dungeon game genre, Three.js as a development platform, and other existing literature. The system design and architecture of the game are given in Chapter 3, which explains the educational

philosophy that underpins the selection of systems and mechanisms as well as the structure of the three-level game. Chapter 4 details the implementation, covering the Three.js scene set-up, the player controller, room generation, question engine, entity management, and user interfaces. The testing and evaluation of the system are reported in Chapter 5. The thesis ends with a summary of findings, a discussion of limitations, and ideas for future research. The complete source code is attached separately.

2. LITERATURE REVIEW AND THEORETICAL BACKGROUND

This chapter reviews the theoretical and empirical studies that served as a foundation for the present work. It starts with a historical overview of educational games and then moves on to game-based learning frameworks, the dungeon game genre, browser-based 3D rendering technologies, existing implementations, and the cognitive science of memory and learning through play. The chapter concludes with the identification of the research gap to be addressed by the thesis.

2.1 Educational Games: Definition and History

Structured play as a pedagogical approach is not nearly a new and modern invention. Board games, simulations, and role-playing have been used for centuries in military training, management education, and language instruction (Prensky, 2007). The invention of digital computers opened up new possibilities: early in the 1970s and 1980s, computer-based learning began to be developed for education. Examples of such software include Oregon Trail (1971) and Where in the World is Carmen Sandiego? A study by Mayer (2019) showed that game mechanics could be used to encourage learning by young people.

The digital educational game (DEG) is a very general term that can be defined as any computer-based game created with the clear purpose of supporting learning outcomes. Some researchers have differentiated between serious games (games constructed with a main purpose other than entertainment) and games where learning is a by-product of the game (Michael & Chen, 2006). In this thesis, the emphasis will be made on the first group a game created to teach specific Computer Science concepts.

Modern definitions focus on the need to make the game's mechanics congruent with the learning goal. Mayer (2019) states that there are two educational game design requirements: it must be entertaining enough to sustain voluntary engagement, and it must

be content-rich in a way that will result in lasting cognitive change. This dual requirement distinguishes serious game design from both pure entertainment design and conventional instructional design. Educational games have experienced significant growth in the market and in academic research over the last 10 years. Literature reviews show that there is consistent evidence of positive impacts on learning outcomes, motivation, and self-efficacy, especially in STEM areas (Clark, Tanner-Smith & Killingsworth, 2016; Camacho-Sánchez et al., 2023). A meta-analysis of 36 studies on DGL in science education was performed by Gyedu et al. (2026) and found a mean effect size of $d = 0.75$ relative to conventional instruction, which is a large effect size according to the standards of education research.

In Computer Science education, the application of a game-based approach has been found in algorithm visualization (Dicheva et al., 2015), programming (Muratet et al., 2020), and data structures (Swacha, 2021). Other research papers also show that game-based approaches can be used in learning network concepts (Katmada, Mavridis & Tsiatsos, 2014). The present work joins this practice by focusing not only on web technologies, but also on data structures, algorithms, networking, databases, and OOP, in a single, consistent game environment.

2.2 Game-Based Learning Frameworks

There are some theories which have been suggested to explain and inform the design of game-based learning experiences. Some of the most widely used theories are GBL theory by Camacho-Sánchez et al. (2023), the Four-Dimensional Framework Theory (4DF) by Lim et al. (2006), and structural theory by Mayer (2019). All these theories have certain similarities, including the role of challenge and feedback, the significance of the learners' agency, and the importance of the narrative context. Figure 1 shows the framework according to Camacho-Sánchez et al. (2015):

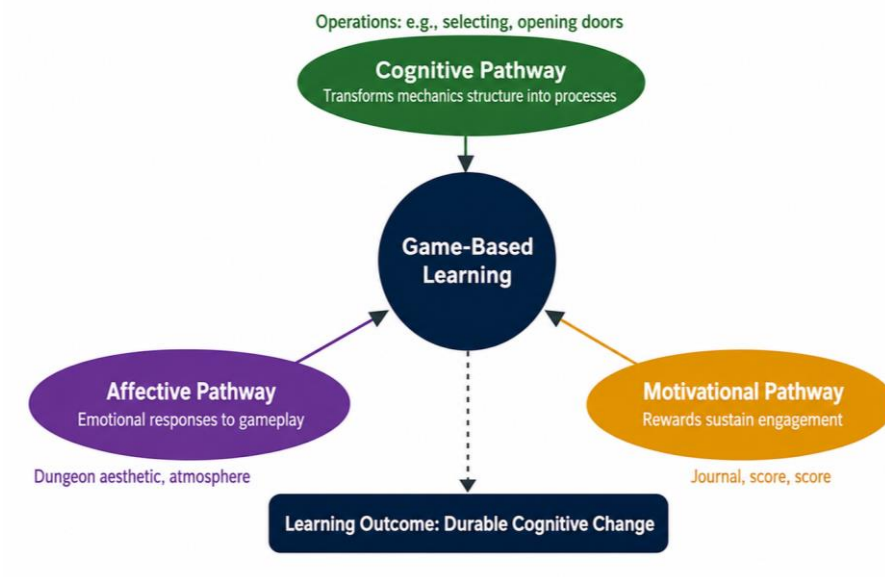


Figure 1: Game-based Learning framework

Camacho-Sánchez et al. (2023) present a cognitive-affective-motivational model that states that game design elements affect learning through three interacting pathways. The cognitive pathway is about the way the game mechanics are designed to process information, the affective pathway is about emotions triggered by the game, and the motivational pathway is about the degree to which game rewards and challenges prolong voluntary participation. The question-and-answer mechanics aim at the cognitive pathway, the dungeon atmosphere and framing the game at the affective pathway, and the experience point system and journal collection mechanics at the motivational pathway. The three-way model is highly applicable to the creation of modern-day games, since in this case it is the question-and-answer aspect which targets the cognitive pathway, dungeon setting and storyline are used for engaging the affective pathway, and experience points and journal mechanics stimulate the motivational pathway.

Mayer's (2019) Cognitive Theory of Multimedia Learning (CTML) also appears in the literature. CTML suggests that learning from combinations of words and images

(multimedia) is more effective than learning from text alone, as long as the capacity of working memory is not exceeded (Mayer, 2019). Multi-mode learning through gaming occurs due to the application of text-based question panels, navigation within three-dimensional space, and visual feedback, including colour-coded correct and incorrect responses, which follow the concepts of CTML (Mayer, 2019). Such ideas were used in relation to a certain state of optimum experience, which makes individuals fully engaged in the action and in full control, with the appropriate degree of difficulty balanced by their skills. Alotaibi (2024) showed empirically that flow-state during gameplay has a high correlation with learning achievements in the educational game context. The difficulty progression of dungeon games is thus proposed to be progressively made based on the various curriculum levels that are encountered as the player gets further into the game.

Gamification is another concept that is associated with gaming but differs from GBL (Deterding et al., 2011). It is the incorporation of surface-level game features (points, badges, leaderboards) into existing educational tasks, while GBL is the embedding of educational content into a complete game environment. Evaluating the full GBL implementations, Swacha (2021) concludes that they are always more successful in terms of engagement depth and knowledge retention than gamification, which is why GBL was chosen for this thesis, and a full game was designed instead of a gamified assessment tool.

A model that is particularly relevant for the present work is the narrative-centered learning environment model offered by Rowe et al. (2011). This is a learning environment built around an interactive narrative that represents the learning content for the entire learning process. The dungeon-exploration story of this game (where the player character gathers bits of knowledge from defeated enemies, unlocks doors by proving knowledge, and builds a scientific journal) illustrates this story-focused approach.

2.3 Dungeon-Style Games as a Genre

Dungeon Games are among the oldest and most consistent genres of gaming. Procedural environment generation is a key feature of dungeon games along with room-by-room exploration, both of which continue to evolve even in modern times (Lazaridis & Fragulis, 2024). Examples of games that have emerged recently include Hades II (2024) and Tainted Grail: The Fall of Avalon (2024), which keep on developing the genre, still holding onto its main features, including rooms' navigation, fighting enemies, loot acquisition, and increasing difficulty levels (Lazaridis & Fragulis, 2024). The most important structural elements of dungeon-type games are directly pertinent to educational design. First, the room-based design gives a natural progression or a unit for the content, as it can be spread across discrete spatial points. Second, whether they're locked with keys, puzzles, or combat, door mechanics provide a meaningful moment in the game where the player must prove their skill before progressing. Third, the enemy encounter system features a challenge framework with context and failure with consequences, but also recovery. Fourth, the loot and inventory systems enable collection behaviors which can be traced to knowledge accumulation goals (Stasiak, 2023).

A number of recent works have been based on dungeon-like style conventions for educational uses. According to Sánchez et al. (2020), the dungeon-escape format is a language-learning method that they report provided greater benefits for vocabulary retention compared to the traditional flashcard study method. The inherent difficulty progression of the genre also fits nicely with Bloom's taxonomy. This is where the difficulty of each room can be increased from recalling information (lower order) to analysis and synthesis (higher order) as the player progresses (Camacho-Sánchez et al., 2023). Figure 2 below shows a representation of the existing Bloom's taxonomy alignment across game levels:

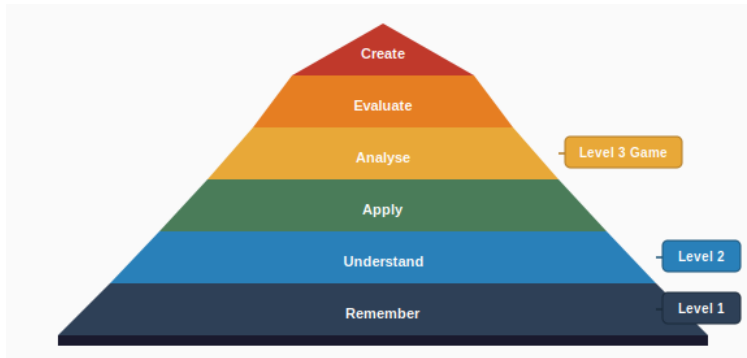


Figure 2: Bloom's taxonomy alignment across game levels (Vanderbilt University Center for Teaching, n.d., via Management and Strategy Institute, 2023)

The increase in difficulty from level to level can be demonstrated using examples. In level 1, a sample question that can be asked is what HTML stands for. In the second level, a sample question can include what HTTPS provides. In the third level, a sample question can be about the sorting algorithm that performs best and why. Unlike classical dungeon games which generally used a top-down or isometric view (Brown et al., 2023), the present thesis takes a first-person perspective in 3D to explore the dungeon. The reason for this design decision is the potential of the first-person perspective in 3D environments created with Three.js to evoke a sense of presence and spatial engagement in browser-based 3D applications (Dirksen, 2023).

2.4 Three.js and Browser-Based 3D Game Development

Three.js is an open-source JavaScript library that simplifies the use of the WebGL API for creating and rendering 3D scenes in the browser (Dirksen, 2023). First launched in 2010, the library has slowly expanded to include a wide range of tools for creating, applying materials, lighting, managing the camera, animating, and post-processing geometry. Three.js is currently the most popular WebGL abstraction library (as of 2024), has more than 100,000 stars on GitHub, and has a wide range of documentation from the community.

Three.js has many benefits over other game development platforms used in education. Three.js applications, as opposed to native game engines (Unity, Unreal Engine),

do not need to be installed and run in any modern web browser on any operating system. The cross-platform accessibility is especially relevant in the context of education in institutions, where students use learning materials on a range of heterogeneous devices (Stern & Shalev-Shwartz, 2020). In addition, the JavaScript world in which Three.js is built is the focus of Computer Science courses, which gives students the chance to analyze and learn from the game's source code.

The technical structure of an application built with Three.js is always built from three major parts: The Camera is responsible for the viewer's angle, the Scene is a hierarchical graph of objects with each element characterized by a Geometry, which defines its shape, and a Material, which defines its appearance. The renderer is the entity that transforms the geometry of the scene into the output of pixels through WebGL. The game loop (which is a repeating render function, called by the requestAnimationFrame function) runs at the display's refresh rate, which allows for smooth, real-time interaction (Dirksen, 2023; Stemkoski & Pascale, 2021).

The Pointer Lock API (W3C standard), which handles the mouse pointer controls for continuous camera rotation, is a necessary addition to Three.js for this type of 3D environment. Stern & Shalev-Shwartz (2020) name the Three.js scene management pattern as the most prevalent architectural pattern for 3D first-person experiences when developed for browser use. In such environments, collision detection is usually carried out by testing axis-aligned bounding boxes (AABB), which is a relatively inexpensive process that is suitable for the regular geometry of rooms in dungeons. However, there are concerns about Three.js performance when using the library for educational purposes. Wimmer and Wonka (2006) list the following factors as the main ones for real-time applications: scene complexity, draw call count, and texture resolution.

2.5 Related Works

The current research shares common ground with several studies mentioned in the recent literature. Sanchez, Young, and Jouneau-Sion (2020) developed an online educational escape room for secondary school science using HTML5 canvas rendering and embedding multiple-choice questions as door unlocking mechanisms. They were found to significantly improve post-test scores compared to a lecture control group, and qualitative data showed that students engaged with the content at a high level. The present thesis builds on this work by working in a fully 3D setting and a more complex knowledge-gathering system.

As mentioned by Muratet et al. (2020), there is the Prog&Play, which is a computer game developed using the spring real-time strategy engine to educate people about programming. In the game, one can code in order to control certain in-game units. Although this approach is entirely different from the current paper, the concept of having code skills demonstrated for success in the game is similar to the current idea. Dicheva et al. (2015) carried out a systematic literature review about gamification in education analyzing 34 publications dated from 2010 till 2015. Among other results obtained from their analysis, there was a problem that can be considered typical for most scientific work in the field – lack of control groups and standardized measurements.

Stasiak (2023) offers a framework for serious game design for higher education that consists of five design principles: realignment of game goals and learning outcomes, meaningful failure states, knowledge scaffolding for progressions, provision of a persistent knowledge repository, and a social or narrative context for content delivery. These principles explicitly shape the current design of the game: questions are pitched at the correct level of difficulty in relation to the level of the curriculum; wrong path rooms are meaningful failure states? The game has three levels of progressive difficulty, thus

providing scaffolding; the in-game journal is the persistent knowledge repository? And the dungeon narrative provides a context for the framing.

Katmada et al. (2014) created an educational game with network concepts by customizing their 2D game engine, and found that players had better knowledge retention than the non-player controls. Their work is one of the few studies that focus on the content of Computer Science networking using game-based learning. This finding also highlights the potential of educational games for transferring networking theory into practice, thereby necessitating additional studies on this aspect.

2.6 Memory, Recall, and Learning through Gameplay

Significant advances have been made in the relationship between gameplay and memory consolidation, and it has been well established in cognitive science (Bipp et al., 2024). In 1885, Ebbinghaus postulated that forgetting happens exponentially over time; this means that games have to maintain memory and recall (Murre & Dros, 2015). Figure 3 below shows Ebbinghaus forgetting curve and game-based countermeasures as proposed by Murre and Dros (2015):

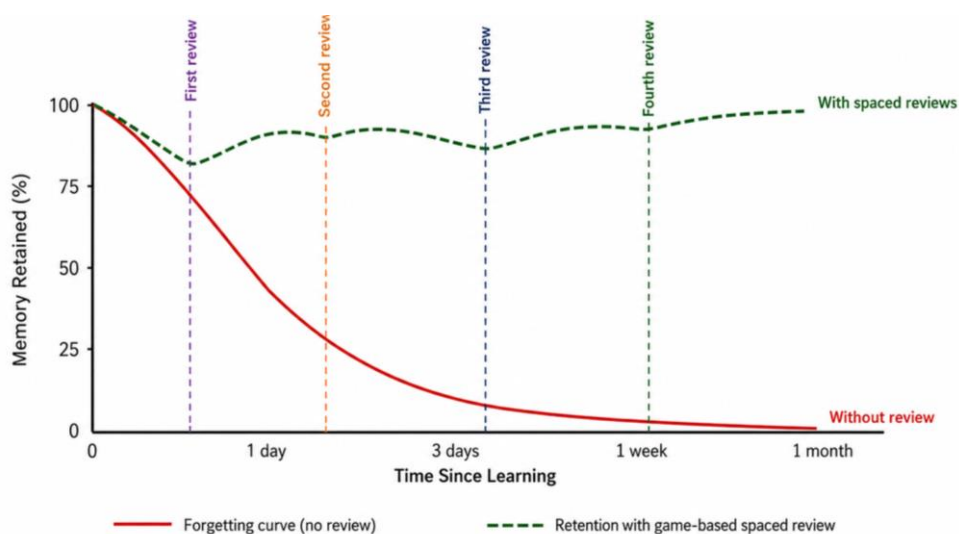


Figure 3: Ebbinghaus forgetting curve and game-based countermeasures

One of the best supported interventions for long-term retention is spaced repetition, which involves re-encountering material at gradually increasing time intervals (Cepeda et al., 2006). Games where knowledge delivery happens in several instances serve as a form of spaced repetition, whether intended by the designer or not. Generate effect, which states that actively generated and retrieved information is better memorized than passively acquired information through reading, is closely linked to the question mechanics that this thesis is concerned with (Walsh et al., 2023). Unlike simple recognition of correct answers, each battle and door-unlock in the game engages players in retrieval and application of the information. Immediately after players make a choice, the game gives feedback, showing both correct answers and brief explanations, thus helping learners to better understand the information they received.

The model that Rajcsanyi-Molnar et al., (2024) suggested describes a relationship between game mechanics and learning effects. This model posits that games affect learning processes by influencing learner behavior. Thus, highly interactive games accompanied by rapid feedback are expected to have better results than less interactive games. It is precisely for this reason that question and answer mechanics, including color-coded feedback and explanations, have been included in this research.

Evidence from cognitive psychology and educational neuroscience also supports the role that 'narrative context' plays in supporting memory and learning. Research indicates that episodic memory systems (Mayer, 2019; Shute, 2011) can enhance memory encoding through stories and spatially-organized environments. The dungeon environment, different rooms, repetitive enemies and exploration-based storyline will create memorable experiences for players that could facilitate recall of question content during later study.

The journal feature described in this thesis was created to aid knowledge consolidation. From most of the already-developed games, the best strategy that can be used

to develop an educational game is for players to gather scrolls, hints, and wall inscriptions during the game. They would also slowly progress into their own unique reference journal that can be consulted at any time during the game. This strategy is based on the idea of elaborative interrogation: that is, reinforcing comprehension through linking new information to old information stored in memory structures (Dunlosky et al., 2013).

2.7 Summary and Research Gap

The literature reviewed in this chapter provides a strong theoretical and empirical background for the current study. The use of games in learning is an organized approach that has been proven to offer benefits and is effective in education, especially in STEM learning. Formative assessment and progressive knowledge acquisition are meaningful to be embedded in a dungeon game by way of a structurally appropriate template. Three.js is a very mature, and pedagogically friendly platform to provide such experiences inside the browser. Hence, the use of a question-and-answer scheme as well as color feedback and explanations in this study will reinforce this interactive way of learning.

This thesis aims to fill the gap of an integrated and coherent Three.js-based educational dungeon game, which combines multiple different knowledge gathering mechanics. It focuses on using question-gated doors, combat encounters, chest rewards, and environmental inscriptions to address the contents of the undergraduate Computer Science curriculum. This is what the current thesis aims to solve by providing a full-fledged design, implementation, and evaluation of such a system.

3. SYSTEM DESIGN AND ARCHITECTURE

This chapter describes the design rationale and architectural decisions made for the dungeon game created for this thesis. The educational philosophy that informed the design decisions is followed by an explanation of the mechanics of each game, and then the design of the system architecture, data flow, and the logic behind the level layout. The chapter sets the conceptual and structural foundation between the theoretical framework discussed in Chapter 2 and the technical implementation discussed in Chapter 4.

3.1 Game Concept and Educational Philosophy

The most important design criterion for this project is that educational content is not a separate entity, but part of the game's main flow. This idea is representative of the difference between intrinsic integration (where the rules of a game are used to teach the learning goal) and extrinsic integration (where educational content is loosely entangled in the game). The game is set in a multi-level dungeon, with the player as a scholar trying to escape a maze of knowledge. This story context is significant for two reasons. Firstly, it makes the question-and-answer actions useful examinations of the player's understanding. Second, it provides a memorable experience and environment that can help facilitate the retention of knowledge as suggested by Mayer (2019). The educational material is geared towards undergraduate Computer Science courses such as Web Technologies, Data Structures, Number Systems, Algorithms, Networking, Object Oriented Programming. These subjects fall under the scope of the Computer Science curriculum guidelines developed by the ACM/IEEE Joint Task Force (ACM & IEEE-CS, 2013).

The game design is also in line with the updated Bloom's taxonomy (Anderson & Krathwohl, 2001). At level 1, the knowledge learned will be limited and basic facts will be recalled like HTML, binary numbers, stacks etc. At level 2, students begin to understand and apply concepts like CSS, binary search complexity and HTTPS. Level 3 focuses on

higher order thinking skills such as analysis and evaluation such as comparing algorithms, SQL statements and object-oriented programming encapsulation. This progression makes it easier to increase the difficulty of the game as it reflects their higher cognitive abilities rather than more content.

The overall design is based on the three principles outlined by Stasiak (2023). First, failure should be meaningful. Wrong answers not only cost a point but also put the player at the intersection of penalty zones, which have darker images and more complicated layouts, thereby reinforcing the penalty for a lack of knowledge. Second, knowledge has to be shareable and malleable. There are various types of educational content throughout the game such as enemy scroll drops, chest hints, wall descriptions and the journal is accessible at all times. Third, assessment should take place instantly, and be a part of the game. Players immediately get feedback on their responses to questions, and short explanations follow to help them understand the correct answers.

3.2 Game Mechanics Overview

The game features six different categories of mechanics, each of which has an educational and engagement purpose. These are described separately in the subsections below.

3.2.1 Room Exploration and Navigation

The dungeon is structured as a series of 3D rooms that are linked together within a 3D scene. The player moves around using a first-person perspective (keyboard and mouse controls), offering a level of spatial immersion that is typical of the dungeon exploration genre. Using first-person perspective is a conscious design decision, with both Dirksen (2023) and Stern and Shalev-Schwartz (2020) showing that the experience in a first-person

perspective on a browser-based environment is more engaging and powerful than the same experience in a top-down presentation.

The levels consist of a main corridor of rooms from spawn to exit, with side rooms branching off with doors. The geometry of the room is designed to be as simple as possible, using axis-aligned box primitives to maintain uniformity in performance on a variety of devices used by students. The ambient lighting is kept low whilst point-light sources are added to the torches on objects at the edges of the room, providing a "dungeon" atmosphere and ensuring that the interactive objects can be seen.

Player movement utilizes axis-aligned bounding box (AABB) collision detection that checks the player's bounding volume against every wall and locked door object in the scene at every frame. In order to ensure seamless movement, the game uses sliding collision detection by first attempting to move on each individual axis before moving using the movement vector itself. This ensures that players will not be trapped inside walls in the maze dungeons.

3.2.2 Question and Answer Door-Unlock System

The main gameplay mechanic of the game is the question-gated door. If the player approaches and triggers an interaction event, each level has at least one exit door that can be unlocked. When interacted with, a question panel is shown for a multiple-choice question selected from the curriculum topic of the level. The question panel is placed over the 3D scene, and temporarily halts the movement of the game world, thereby maintaining the spatial context in the background.

Clicking on an option will provide instant feedback. Here, the correct answer will be marked with a green background, the incorrect answer (where relevant) will be marked with a red background, and a text block with an explanation of the answer will appear beneath the options. This immediate, elaborative feedback is directly in line with the

principles of the generate effect outlined in Section 2.6, meaning that even if students are wrong, they still have a learning moment, instead of simply losing a point.

If the player answers correctly, the exit door will do an unlock animation (change color from crimson to green and raise out of the doorway) to allow for entry into the next room or level. If the player answers wrong, a different door is unlocked (the "penalty room" door), and the player loses one health point. This visual distinction between the right and wrong-path door, both in the question panel and in the 3D room, helps to emphasize the spatial and narrative nature of the wrong-path recall.

3.2.3 Wrong-Answer Penalty Paths

Penalty rooms are separate sub-areas with a separate entrance through a wrong-path door. Their walls are rendered in deep crimson, indicating danger and disorientation. There are no enemies, chests, or inscriptions in penalty rooms, and the player is not moving closer to the level exit. The player is then required to go back to the main corridor and try the question again. This mechanic is an example of the meaningful failure principle that Stasiak (2023) called out and which was designed as noted by Shute (2011) for stealth assessment. Here, failure is taken as an in-world event and not a numerical deduction, placing the assessment experience within the game narrative. The health penalty for wrong path entry (one heart per incorrect answer to the door) adds a game management element that encourages players not to overlook recalling the correct answer but does not diminish the impact of any single wrong answer.

3.2.4 Enemy Encounters and Scroll-Drop System

Several enemy entities are designed and depicted as crimson-colored stylized 3D humanoids within each level. Players start attacking enemies when they get too close and press the 'E' interaction key. When a player meets an enemy, a question panel is shown,

which contains a curriculum question that is linked to that particular enemy. This design has the formative assessment events spread throughout the game world, which gives the pacing structure of assessment being scattered throughout the game and not just at one gate.

When the enemy is defeated (by choosing the right answer), they die and drop a scroll item. When the dropped scroll is collected, it is placed in a player's journal. Each scroll is an elongated informational text about a topic on the curriculum, which serves as a reward for the successful knowledge demonstration. The scroll mechanism means that no enemy is defeated without the collection of a knowledge artifact. If they are defeated, they will generate a knowledge artifact.

In combat, a wrong answer results in the player losing a health point and the question panel closing. The player can re-engage the enemy in attempt to answer correctly. This style of re-engagement allows combat not to be a one-shot, high-stakes test, alleviating the pressure and keeping the motivation to show that one knows what's right. Nevertheless, it has been recognized that failure experience can contribute to negative emotions, including frustration and anxiety among certain learners (Conati & Maclaren, 2009).

3.2.5 Hidden Chests & Wall Inscriptions

There are two further sources of knowledge delivery throughout the rooms of each level: treasure chests and wall inscriptions. Treasure chests are three-dimensional objects with a distinctive gold trim and a light source that pulses. When interacted with, a chest will open up with an animation and show a hint panel with a short mnemonic/explanatory note about a curriculum subject. The hint is automatically placed in the player's journal and can be seen at any time.

Wall inscriptions are two-dimensional planar objects mounted at eye level, and rendered by a CanvasTexture function with typeset text on a stone plaque background. Each inscription has a short quotation or fact related to Computer Science. When it interacts with

other inscription(s), its full text will appear in a panel and be recorded in the journal. Unlike questions-and hints, wall inscriptions can be given in a non-evaluative manner, that is, they provide knowledge without evaluation, which is consistent with the pre-learning and consolidation role highlighted by the spaced repetition literature (Cepeda et al., 2006). The spatial layout of chests and inscriptions around the room is intentional. It rewards exploration rather than traveling along one critical path. The design decision was made on the basis of the motivational principle of optional depth: if a player is willing to go beyond the minimum amount of required path, they will get additional learning content as a reward for exhibiting curiosity-driven behavior.

3.2.6 Player Journal System

The player journal is an in-game data store that can be accessed at any time with the J key. It displays knowledge pieces that have been gathered and grouped into 3 tabs: Scrolls (from enemies defeated), Hints (from opened chests), and Inscriptions (from wall plaques). The publications are presented by title, topic, and full text. The journal has several educational purposes. It can be used as a reference guide to get a quick preview of what a player has already seen before they are presented with a question-gated door, which meets just-in-time retrieval practice. It is a collection system, which makes learning to collect knowledge fun. The list of entries is displayed in the HUD counter and gives a visual cue to how far one has gone with learning. It can serve as a revision structure that summarizes all educational content that has been faced in the play session and can be seen after the play session is over.

The journal design follows the elaborative interrogation principle (Dunlosky et al., 2013). The focus here is not only on giving players correct answers, but also on giving them the context for their answers, so that they can make the connection between their new knowledge and their current knowledge structures. This is further enhanced by the

organization of the journal entries by category of topic, with items grouped by the same headings, thus supporting schema formation.

3.3 System Architecture

The system is deployed as a stand-alone web app that does not require any server-side components. Game logic, rendering, educational content, and game state management are all run in the client browser. The architecture consists of six logical layers, which are described in the following subsections and summarized in Table 1.

Layers	Description
Presentation Layer	Handles all things that can be seen by the player, starting from the layout of the web page that was created using HTML5 and ending with the visual representation, animation, and rendering through Three.js library. Also, this module includes all UI elements like questions, journal, treasure chest, and HUD.
Rendering Layer	Responsible for rendering and manipulation of the 3D environment. It involves the use of the Three.js render engine, scene handling, camera manipulation, shadow maps, light effects, and rendering textures for wall writings.
Game Logic Layer	This layer mainly controls the main game mechanics. This is where the player's movement, collision detection, camera manipulation through the use of Pointer Lock API, enemy, and object manipulation are done via requestAnimationFrame. Room generation, door operation, and the update loop are also executed continuously via requestAnimationFrame.
State Management	State Management stores and updates important information that affects the game, such as player health, location, opened doors, acquired journal pages, scores, interface state, and level transitions.
Content Layer	Contains all teaching material and interactive material used while playing, such as questions in the Quiz, lore scrolls, hint messages and inscriptions, which can be found in different levels.
Delivery Layer	Describes details of local operation of the project. This game can be hosted via local web servers like Python HTTP Server, node.js Serve, and VS Code Live Server. Third-party libraries like Three.js are used via the Content Delivery Network (CDN), so there's no additional build steps involved.

Table 1: System architecture layers and their components

The absence of a server-side component is a deliberate architectural decision. It maximizes deployment accessibility. This means that the game can be served from any static file host, a USB drive, or a local development server, and eliminates the dependency on

network connectivity during gameplay after the initial Three.js CDN load. This aligns with the accessibility requirements of institutional educational deployment contexts discussed in Section 2.4.

The three-dimensional scene is managed as a Three.js Scene graph. Room geometry, entity meshes, and lighting are all added as child objects of the root scene. Room groups are positioned using absolute world-space coordinates, with each level defining room positions relative to the origin. Entity objects are positioned within the same world space, enabling straightforward distance calculations for proximity detection.

3.4 Data Flow and State Management

The game's state is managed through a centralized GameState object that serves as the single source of truth for all runtime data. This pattern, analogous to the flux/redux architecture pattern common in React applications, ensures that all game systems read from and write to a consistent, predictable data structure, eliminating the class of bugs that arise from distributed mutable state.

Table 2 below illustrates the main data flows during a typical gameplay session. The game loop, executing at the display refresh rate via the requestAnimationFrame function, performs three categories of operations on each frame. The first is input processing (reading keyboard and mouse state), second is physics update (player movement and collision resolution), and finally render (Three.js scene draw call). Interaction events, which are triggered by the E key when the player is within range of an interactive object, break this loop to invoke the question or discovery subsystem, which updates the GameState and re-renders the relevant UI panel before returning control to the game loop.

Event	State Change	UI Changes
Correct door answer	Door unlocked in state variable; correctAnswers++	Correct door opens (green color); message displayed

Wrong door guess	Decrease player's health; unlock wrong path door	Health loss animation; wrong door opens (red color)
Defeat enemy	enemiesDefeated++; add scroll to journal	Enemy disappears
Open chest	chestsOpened++; add hint to journal	Lift lid animation; chest panel shown
Read wall inscription	Add inscription to journal	Wall inscription panel shown
Enter portal	currentLevel++; clear and rebuild scene	Level complete message; Clear and rebuild current scene
Player health = 0	gameOver flag set	Game over panel displayed; pointer lock released

Table 2: Principal game events and their state and UI consequences

Table 2 describes the key game events together with respective state transitions and UI actions. There's also a door unlock registry in the GameState object, which is an array of door identifiers that have been unlocked during the session. This registry is accessed by the collision detection routine in each frame: locked doors are considered solid wall geometry, unlocked doors are not called by the collision system, so the player can walk through them. This way, the need for modifying the scene geometry when doors unlock is eliminated. A scene rebuild procedure is used to perform level transitions, which removes all non-persistent scene objects, reinitializes entity lists, resets the door unlock registry, and rebuilds the room geometry and entity population for the target level. GameState journal, score counters, and player health are clearly saved when switching to new levels and carried forward to the next one.

3.5 Level and Room Design Logic

There are three levels, each corresponding to a different stage of a dungeon, where the room layout and entity configuration are also different. Each level is designed to meet the progressive difficulty as described in Section 3.1, which increases in terms of difficulty of questions, number of rooms, and density of entities from Level 1 to Level 3.

Level	Rooms	Enemies	Chests	Topics
-------	-------	---------	--------	--------

Level 1	4 (spawn, main, wrong-path, exit)	2	2	Web technologies, data structures, binary
Level 2	5 (spawn, main, east wrong-path, west bonus, exit)	2	2	CSS, algorithms, networking
Level 3	5 (spawn, main, east wrong-path, west bonus, exit)	3	2	OOP, databases, algorithms

Table 3: Level composition summary (Source: author)

There are common elements found in each level of spatial arrangement, such as a spawn area, a main challenge area, additional side areas, and an exit area. In the spawn area, the player will start playing and can find chests and wall inscriptions that will inform players prior to their exploration. In the main challenge area, there are enemies and a door guarded by questions. The side areas provide some other paths for exploring, while the exit area has the portal needed to complete a particular level. These similarities make navigating easier because the dungeon layout becomes easy to understand. Thus, less cognitive load is used for solving puzzles.

The size of rooms is specified as 'world-space' units? Level 3 rooms are slightly larger than Level 1 rooms, to accommodate the higher density of entities in those rooms. A common geometric construction method is used throughout the rooms: floor plane, ceiling plane, and segmented wall geometry with optional door openings, with width, depth, height, and door configuration flags. This serves as a procedural method to make the room geometry consistent and collision-safe in all levels.

An exit portal on each level is located at the far end of the exit room and is visually marked with a cylindrical mesh, a translucent purple material, and a point light source. The portal is only accessible if the exit door is unlocked with the correct answer to the main question for the level, which means that the exit event must be followed by a successful knowledge demonstration.

There is a clear distinction between wrong path rooms and correct path rooms because wrong paths have walls painted in dark red (Hex #3d0505 compared to #2e2010). This chromatic difference gives a player instant feedback on how far they have gone before they enter the room containing a wrong path, which creates a link between the visual setting and the outcome of recalling the wrong path. The rooms are structurally symmetrical, meaning that they are the same size in terms of their dimensions along the floor plan, so the wrong-path experience is one of navigational 'inconvenience' rather than 'puzzle'.

4. IMPLEMENTATION

This chapter presents the technical realization of the educational dungeon game. It includes an explanation of the development environment, the 3D scene configuration, the player controller, the room construction, the question engine, the entity system, the journal, the state management, and the UI. Structural diagrams and simplified pseudocode are used where appropriate to explain the key logic, but do not show the full source code.

4.1 Development Environment and Tools

The game was built with the aim of being a standalone web application with no build pipeline or server infrastructure required. The complete tool set is listed in Table 4.

Material / Tool / Technology	Version / Standard	Responsibility
Three.js	r128	3D scene rendering using WebGL
JavaScript (ES Modules)	ES2020	Core game logic, state management, and UI control
HTML5	Living Standard	Creation of the player interface, document structure, canvas, and overlay panels
CSS3	Level 3	Visual styling, transitions, and responsive layout
Pointer Lock API	W3C Standard	Mouse capture for first-person camera rotation
CanvasTexture (Three.js)	r128	Procedural wall inscription rendering
Python http.server	3.x	Local static file server for development

VS Code + Live Server	1.88+	Development environment, editor, and hot-reload development server
-----------------------	-------	--

Table 4: Development tools and technologies (Source: author)

The selection of three.js r128 was the best choice because this particular version is the most compatible version among all. This means that the library gets dynamically loaded via CDN URLs without the need for installation on a local machine. The entire remainder of the code is written in vanilla JS (no transpiler/bundler whatever), using ES Module syntax. This choice keeps the project fully transparent and suitable as a learning artifact for students examining the source code.

4.2 Three.js Scene Setup: Renderer, Camera, and Lighting

The Three.js scene creation utilizes three key elements, which are the WebGLRenderer, a PerspectiveCamera, and a Scene. The important configuration parameters are summarized in Table 5 and a code listing of the Three.js renderer, scene, camera, and fog initialization are shown in the code listing 4.1 that follows:

Object	Parameter	Value and Rationale
WebGLRenderer	antialias	true: removes sharpness on room geometry
WebGLRenderer	shadowMap.type	PCFSoftShadowMap: makes smooth shadows
WebGLRenderer	pixelRatio	min(devicePixelRatio, 2): limits GPU load on HiDPI
PerspectiveCamera	fov	72 degrees: wide enough to explore dungeons
PerspectiveCamera	near / far	0.1 / 80: clips geometry outside dungeon bounds
Scene	fog	Fog(0x040201, 6, 28): atmospheric distance fade
AmbientLight	intensity	0.35: minimum base illumination
PointLight (torch)	intensity / distance	0.9 / 9: localised warm torch glow

Table 5: Core Three.js scene configuration parameters (Source: author)

```

const renderer = new THREE.WebGLRenderer({ canvas: document.getElementById('game-canvas'), antialias: true });
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2));
renderer.shadowMap.enabled = true;
renderer.setClearColor(0x040201);

window.addEventListener('resize', () => {
  camera.aspect = window.innerWidth/window.innerHeight;
  camera.updateProjectionMatrix();
  renderer.setSize(window.innerWidth, window.innerHeight);
});

const scene = new THREE.Scene();
scene.fog = new THREE.Fog(0x040201, 6, 28);

const camera = new THREE.PerspectiveCamera(72, window.innerWidth/window.innerHeight, 0.1, 80);
camera.position.set(0, 1.65, 4);

scene.add(new THREE.AmbientLight(0x110900, 0.35));

```

Code Listing 4.1: Initialization of Three.js Renderer, Scene, Camera and Fog.

Every room is built as a Three.js Group object and positioned in world space. Group objects ensure that the contents of rooms remain logically organized and that a whole room can be removed cleanly in level transitions using iteration on the components in the room. Fog is added at the scene level, giving an atmospheric visual effect and restricting draw distance for lower specification devices.

Lighting is done in two phases. A single low-intensity AmbientLight will give a minimum base illumination so that no surface will be completely black. Individual PointLight objects are attached to torch positions inside each room, giving localized warm colored lighting. SphereGeometry meshes are used to animate the torches, with their y-position and scale modulated by sinusoidal functions to produce the flickering effect.

4.3 Character Controller and Movement

The player controller is responsible for keyboard controls, mouse rotations, AABB collision detection, and interaction triggers. The rotation of the camera is controlled by two float variables (yaw and pitch) that are modified by mouse movement events received from the Pointer Lock API on each frame. These values are used directly as Euler rotations for the camera, avoiding gimbal lock when using a combination of horizontal and vertical look.

Motion is accomplished by creating a directional vector from the key press on the keyboard, normalizing it, and finally multiplying it by the speed of motion times the frame delta time. This is very important for smooth motion independent of the frame rate, and this is particularly needed when deploying to other devices with different performance capabilities. The direction vector is first rotated by the current yaw angle to get into the world space before being added to the camera position. When AABB collision is enabled, a bounding box is created around the new position of the player and compared to the bounding boxes of the wall objects and locked door objects in the scene. If the two move vectors get closer to each other, the system tries with just the X component, then with the Z component only. The code below demonstrates this functionality:

```
const fwd = new THREE.Vector3(-Math.sin(yaw),0,-Math.cos(yaw));
const right= new THREE.Vector3( Math.cos(yaw),0,-Math.sin(yaw));
const dir  = new THREE.Vector3();

if (keys['KeyW']||keys['ArrowUp'])   dir.add(fwd);
if (keys['KeyS']||keys['ArrowDown']) dir.sub(fwd);
if (keys['KeyA']||keys['ArrowLeft']) dir.sub(right);
if (keys['KeyD']||keys['ArrowRight']) dir.add(right);

if (dir.length()>0) {
  dir.normalize().multiplyScalar(5.0*delta);
  const newPos = camera.position.clone().add(dir);
  newPos.y=HEIGHT;

  const walls=[];
  scene.traverse(o=>{ if(o.userData&&(o.userData.type==='wall' || o.userData.type==='door')) walls.push(o); });

  if (!collides(newPos,walls)) camera.position.copy(newPos);
  else {
    const px=new THREE.Vector3(newPos.x,HEIGHT,camera.position.z);
    if(!collides(px,walls)) camera.position.copy(px);
    else {
      const pz=new THREE.Vector3(camera.position.x,HEIGHT,newPos.z);
      if(!collides(pz,walls)) camera.position.copy(pz);
    }
  }
}
```

Code Listing 4.2: Delta-Time Movement and Axis-Separated AABB Collision Detection

```

function collides(pos, walls) {
  const pBox = new THREE.Box3(
    new THREE.Vector3(pos.x - 0.3, pos.y - 1.6, pos.z - 0.3),
    new THREE.Vector3(pos.x + 0.3, pos.y + 0.1, pos.z + 0.3)
  );
  for (const w of walls) {
    if (!w.geometry) continue;
    if (w.userData.type === 'door' && !w.userData.locked) continue;
    const b = new THREE.Box3().setFromObject(w);
    if (pBox.intersectsBox(b)) return true;
  }
  return false;
}

```

Code Listing 4.3: AABB detection of collision

This axis-separated resolution gives the wall sliding effect, which prevents the player from getting stuck if going against a wall corner diagonally.

During motion, a camera bob effect is used. The y-position of the camera is modulated sinusoidally by a time accumulator, which increments only when movement keys are pressed. If there is no movement, then during the still-frame, the y-position of the camera is linearly interpolated back to the normal eye height, resulting in a smooth transition. The flowchart below represents a picture of the main game loop:

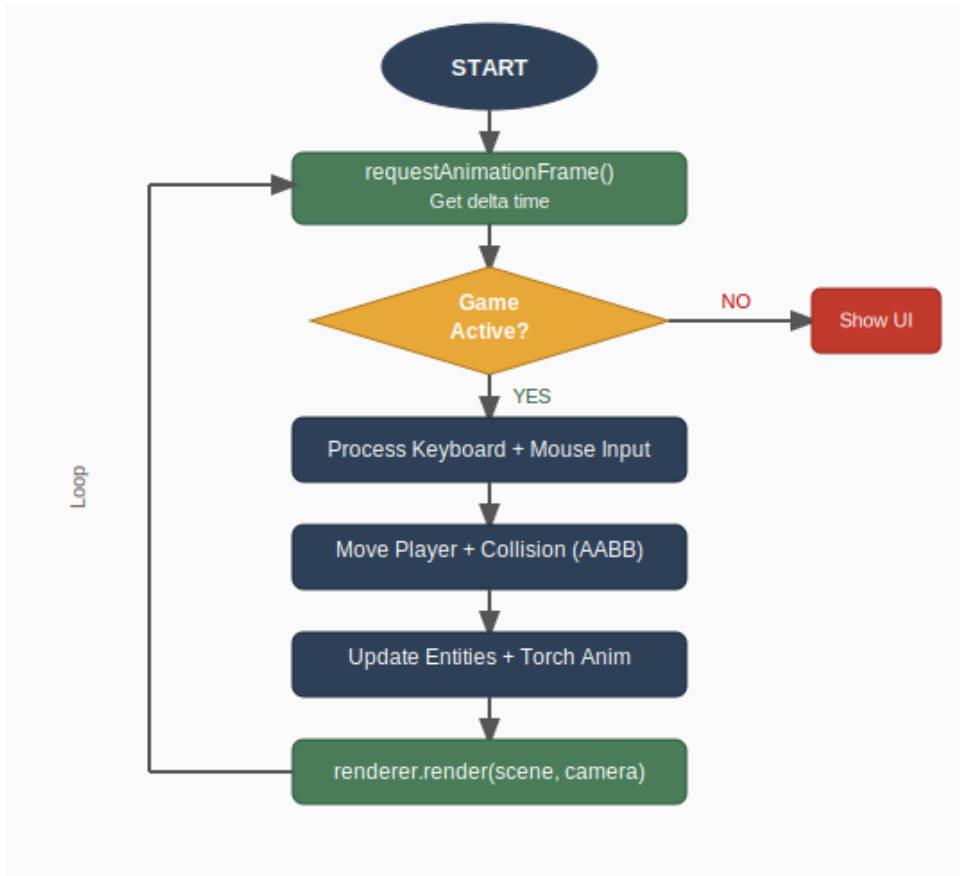


Figure 4: Main Game loop

4.4 Room Generation and Dungeon Layout

A room-construction method is provided for each room in the level layout. Each level is individually built by a builder function invoking a room-construction method. The room method takes in a configuration object that defines position, dimensions and door flags. Three.js BoxGeometry primitives are used to create wall geometry. If a wall face is to be opened for a door, it is divided into three sections: a left side, a right side and a top lintel. This is done in segments, making the opening geometry a precise one without having to use Boolean subtraction, which is not natively supported by Three.js.

The door objects are created as individual BoxGeometry meshes with a red color (locked) and placed in the door opening. Each door is provided with a user Data object, which consists of the door's name, locked/unlocked status, and orientation. The collision system checks the locked parameter for each door to determine whether it is collidable or

not. When a door is unlocked, the locked parameter is reset, and the door mesh animates itself to move upwards from the lintel.

Room Parameter	Type	Description
x, z	Float	World-space positioning in the room
width, depth, height	Float	Room Dimensions in Three.js world coordinates
hasNorthDoor, etc	Boolean	Flags indicating presence of doors on particular wall faces
northDoorId, eastDoorId	String	Identifier for each door instance
isWrongPath	Boolean	Specifies crimson wall color for rooms with incorrect paths
isExit	Boolean	Creates the exit portal that completes the level

Table 6: Room configuration parameters (Source: author)

All room positions are stated in a flat world coordinate system. Room arrangement is done in accordance with the z-axis, whereas side rooms are arranged parallel to the x-axis. Such an arrangement makes the entire building look like a comb structure, which is repeated throughout the three floors. This allows the player to build a transferable spatial mental model throughout the game.

4.5 Question Engine: Loading, Evaluation, and Feedback

The question engine is structured around a static array of questions that is defined in the content module. Each question object is composed of a text for the question, four option strings and the index of the right answer, an explanation string and a label for the topic. Questions are associated with specific game objects at level-load time by storing a question identifier in each entity's user Data object. Below is an image showing the sample level 1 question followed by a wall inscription:

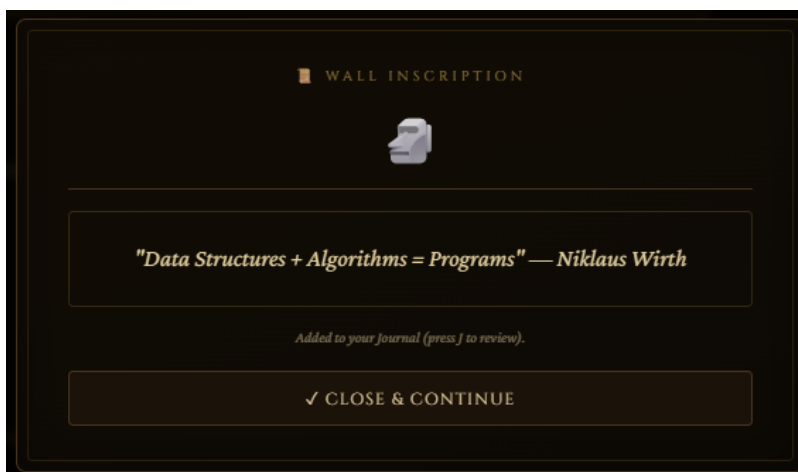
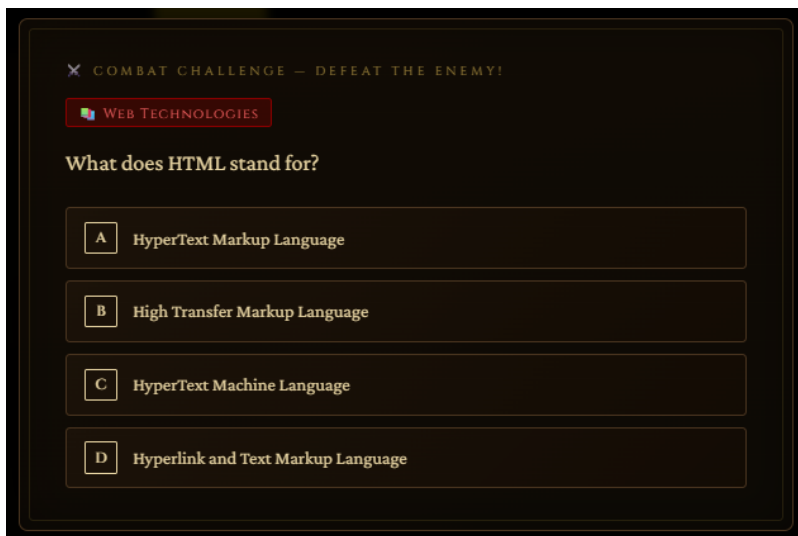


Figure 5: Level 1 question followed by a wall inscription

If an event is triggered, the engine fetches the question object for that event using the identifier and populates the HTML question panel. Four option buttons are dynamically generated and added to the options container. Each button registers a single-use click handler. In case there is a click event from any button, the function will validate whether the selected index matches the correct one and add classes 'correct' or 'incorrect' to the button. The function will also show the explanation for the respective selected index, update scores, and finally close the modal window after 2.6 seconds.

The 2.6-second period is chosen intentionally. According to the results obtained by Shute (2011), elaborative feedback requires some time to process. In case the player was supposed to close the panel immediately after grabbing it, it would be impossible for him to have enough time to read the explanation. Once any button is clicked, it cannot be used again, preventing possible multiple selection. While the question screen is showing, the main game loop does not pause, because Three.js keeps rendering the scene in the background. Below is a flow representation of the question engine flow:

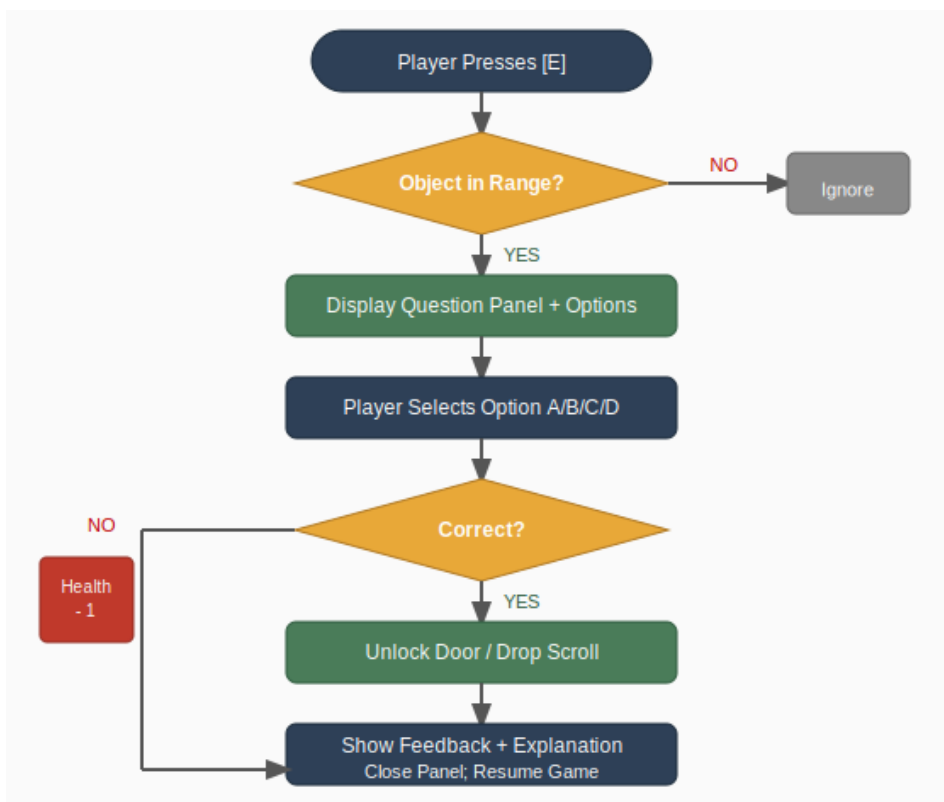


Figure 6: Question engine flow

4.6 Enemy System and Scroll-Drop Mechanism

The enemy entities are constructed from Three.js Group objects, which consist of a body mesh, a head mesh, meshes for horns, eye meshes, and finally, a floating gold cylinder (scroll indicator). This way, all child meshes will be created in the group, and they can be animated as a single entity. The userData object of each enemy group includes an enemy identifier, question identifier, scroll identifier, alive flag and a patrol time accumulator. The

patrol animation is organized so that it rotates the group's y-rotation slowly each frame, using a sinusoidal function of the patrol time accumulator, so that living enemies will rotate slightly, whilst the room geometry in that area remains static. The y-position of the floating scroll indicator also oscillates to indicate the enemy is carrying a collectible item.

When the enemy gets defeated, the alive flag is deactivated, and a sinking animation is triggered. The y-position of the enemy group will be reduced by one interval callback until it is below the floor, then it is deleted from the scene and from the active enemies list. The scroll item is read from the content database to be added to the GameState journal. A pop-up notification appears in the HUD to notify of the addition.

4.7 Chest Interaction and Wall-Text Rendering

Chest entities are a group of objects that consist of a base box, a lid box, a trim strip, and a point light. A separate userData marker is found on the lid, identifying it as the animated component. When the flag is opened, it is set and a setInterval animation increments the x-rotation of the lid while modulating its y-position with a sine function, creating a realistic lid open arc over about 50 frames. The brightness of the point light is increased when the target is opened, providing a reward glow. The inscriptions on the walls are rendered differently. Each inscription is made up of a PlaneGeometry object, and the texture material of the object is generated through the rendering process from a CanvasTexture object created at runtime. For JavaScript, a Canvas element having 512 x 256 pixels is generated with a background color of a dark stone color, and text is written onto it in gold strokes using the Canvas 2D API. The canvas snapshot code is shown below:

```

ctx.fillStyle = '■#2a1f0e';
ctx.fillRect(0, 0, 512, 256);
ctx.strokeStyle = '■#8b6914';
ctx.lineWidth = 8;
ctx.strokeRect(4, 4, 504, 248);

const text = WALL_TEXTS[cfg.textIndex % WALL_TEXTS.length];
ctx.fillStyle = '■#f0d080';
ctx.font = 'bold 18px serif';
ctx.textAlign = 'center';

const words = text.split(' ');
let line = '', lines = [], maxW = 460;
words.forEach(w => {
  const test = line ? line + ' ' + w : w;
  if (ctx.measureText(test).width > maxW) { lines.push(line); line = w; }
  else line = test;
});
lines.push(line);

const startY = 128 - (lines.length * 24) / 2;
lines.forEach((l, i) => ctx.fillText(l, 256, startY + i * 28));

const texture = new THREE.CanvasTexture(canvas);
const plaque = new THREE.Mesh(
  new THREE.PlaneGeometry(2.5, 1.2),
  new THREE.MeshBasicMaterial({ map: texture })
);

```

Code Listing 4.7: Wall inscription displayed as CanvasTexture

Canvas gets converted into a Three.js texture and used in conjunction with the plane mesh with the help of the MeshBasicMaterial, which always renders the texture at full brightness regardless of the situation in the scene. In this manner, any textual content can be shown on an in-game surface without using any additional fonts or textures.

4.8 Journal UI: Collection and Display

The journal is applied as a static, HTML panel that is placed over the Three.js canvas. By default, it's hidden and can be switched on with the J key. The panel has three-tab buttons, and three content divisions, each representing a journal category: Scrolls, Hints,

and Inscriptions. A click handler eliminates the active class from all tabs and content divisions and adds it to the selected pair of tabs and content divisions.

When the panel is opened, the three arrays in the GameState journal object are run through and an HTML string is created for each entry, which is rendered as a journal entry. These listings are rendered in card format, displaying the title, the Topic Badge (if applicable), and the complete content of the item. The render method is responsible for updating the innerHTML of the content div, so the rendering will reflect the current state. A small entry counter badge on the journal allows the HUD to see the journal at all times. This long-lived visibility acts as a reminder of the collection of knowledge and strengthens the motivational aspect of the accumulation mechanic outlined in Section 3.2.6. The proximity interaction flow is represented in the paragraph below:

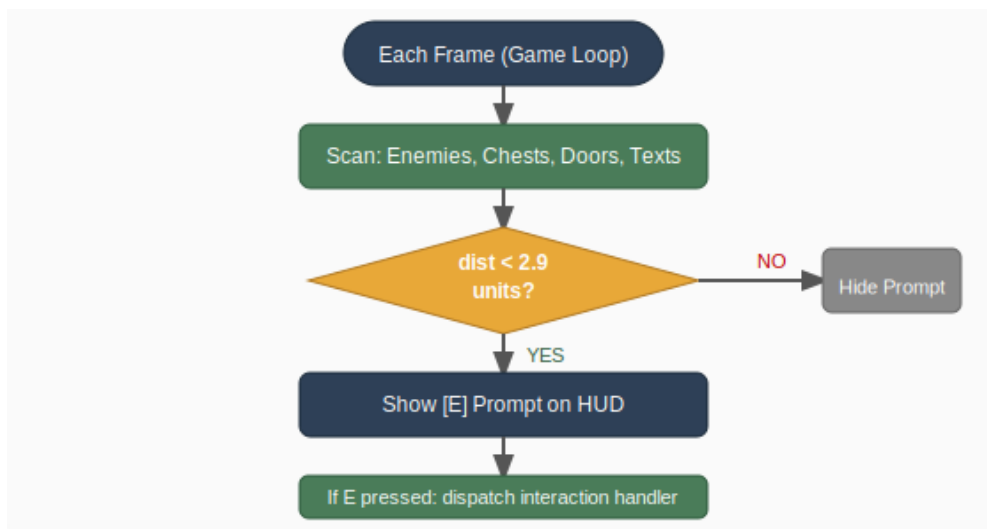


Figure 7: Proximity interaction system

4.9 Game State Management

All game sub-systems refer to a single JavaScript object, GameState, which contains the state of the game. Those sections are logical and consist of player stats (health, position, collected items), dungeon progression (current level, unlocked doors), journal (scrolled, hints, inscriptions), UI flags (question active, journal open, game over, game won), and score counters (correct answers, wrong answers, enemies defeated, chests opened). The

most significant of these are the UI flags. These are being checked at the beginning of each iteration of the game loop. When any blocking flag is set, Movement/Interaction processing is not performed at all. This means they can't move while a question panel is up, or set up a new interaction while a chest panel remains open. These flags can either be stripped using the close icons available in the panel or through a timeout after the feedback to the question is provided. Once the player reaches the exit portal, a transition will be performed between levels within the game. Transition involves the resetting of all the objects in the current scene, resetting of the door unlock registry and also rebuilding of the entire scene. Journal, score counters, and a player's health are maintained. The player is repositioned at the spawn point of the new level and the yaw is reset to face into the dungeon.

```
const GameState = {
  player: { health: 3, maxHealth: 3, position: { x: 0, y: 0, z: 0 }, rotation: 0, collected Scrolls: [], collected_chests: [], collected_hints: [] },
  dungeon: { currentLevel: 1, maxLevels: 3, currentRoom: 0, rooms: [], doorsUnlocked: [], wrongPathVisited: false },
  journal: { scrolls: [], hints: [], wallTexts: [], isOpen: false },
  ui: { questionActive: false, chestActive: false, wallTextActive: false, journalOpen: false, gameOver: false, gameWon: false, paused: false },
  score: { correctAnswers: 0, wrongAnswers: 0, enemiesDefeated: 0, chestsOpened: 0 },
  reset() {
    this.player.health = 3; this.player.maxHealth = 3;
    this.player.position = { x: 0, y: 0, z: 0 }; this.player.rotation = 0;
    this.player.collected Scrolls = []; this.player.collected_chests = []; this.player.collected_hints = [];
    this.dungeon.currentLevel = 1; this.dungeon.currentRoom = 0;
    this.dungeon.rooms = []; this.dungeon.doorsUnlocked = []; this.dungeon.wrongPathVisited = false;
    this.journal.scrolls = []; this.journal.hints = []; this.journal.wallTexts = [];
    this.ui.questionActive = false; this.ui.chestActive = false; this.ui.wallTextActive = false;
    this.ui.journalOpen = false; this.ui.gameOver = false; this.ui.gameWon = false;
    this.score.correctAnswers = 0; this.score.wrongAnswers = 0;
    this.score.enemiesDefeated = 0; this.score.chestsOpened = 0;
  },
  addScroll(s) { if (!this.journal.scrolls.find(x => x.id === s.id)) { this.journal.scrolls.push(s); return true; } return false; },
  addHint(h) { if (!this.journal.hints.find(x => x.id === h.id)) { this.journal.hints.push(h); return true; } return false; },
  addWallText(t) { if (!this.journal.wallTexts.includes(t)) { this.journal.wallTexts.push(t); return true; } return false; },
  takeDamage() { this.player.health = Math.max(0, this.player.health - 1); if (this.player.health <= 0) this.ui.gameOver = true; return this.player.health; },
  unlockDoor(id) { if (!this.dungeon.doorsUnlocked.includes(id)) this.dungeon.doorsUnlocked.push(id); },
  isDoorUnlocked(id) { return this.dungeon.doorsUnlocked.includes(id); }
};
```

Code Listing 4.8: The structure for the singleton GameState function

4.10 UI/UX Design: Menus, HUD, and Overlays

The user interface consists of three layers that sit on top of the Three.js canvas. The bottom layer is the canvas itself. Always visible HUD elements are located in the middle layer: health hearts display, level indicator, score counter, and journal entry count badge. The upper layer has modal overlay panels that are displayed as a result of actions occurring

in the game. Below is a demonstration of the sample room UI interface seen in level 1 of the game, showing the different elements as described in the implementation.



Figure 8: Room UI interface seen in level 1 of the game

The HUD is created using fixed-position HTML elements at each corner of the view. The health display takes the form of heart emojis, and its font size is set at 18pt; the full hearts will denote the number of lives remaining, while the empty hearts will denote the number of lives lost. Another function has been created to refresh the HUD anytime there is an event that changes the game state, hence the absence of a loop for continuously checking. Creating a modal panel involves setting the property being displayed to flex after one animation frame, followed by the addition of the visible class, which make the panel to fade in. This double action will ensure that the CSS transition will work properly across all browsers. Panels are closed by reversing the transition then setting display to none after a timeout of 350ms, equal to the transition time. This stops the sudden disappearance which would happen if the display was set to none.

The visual style is dark, dungeon-themed, and restrained with minimal colors. Near-black values (050505 - 2e2010) are used for backgrounds. Warm gold (c9a84c, ffd700) is

used for accent colors (in UI elements), crimson (8b0000) for danger and locked doors, and emerald green (228b22) for unlocked doors and correct answers. For decorative headings, the Cinzel and Crimson Pro font families downloaded from Google Fonts are used, and Arial is used for body text. The start screen does not have a start button, but instead has a prompt for the Pointer Lock. Clicking the canvas asks for pointer lock, which results in a native browser confirmation dialog, and then starts capturing mouse input. An overlay with game title, control instructions, and a styled enter button is displayed until pointer lock is acquired, after which it is hidden. This is a standard procedure for first-person applications in browsers (Dirksen, 2023).

4.11 Challenges Encountered and Solutions

During development a number of technical and design issues arose. This section records only the most important and the solutions used.

Challenge 1: ES Module imports from the file:// protocol.

Browsers block ES Module imports when HTML files are opened directly from the file system using the file:// protocol. This made it impossible to test without using a web server. The solution is to serve the project using any static HTTP server, such as Python's built-in `http.server` module. For instance, a local server may be started from a console using the commands `cd path/DungeonGame python -m http.server 8000`.

Challenge 2: Smooth movement regardless of the frame rate.

The first testing showed that devices can vary in how fast they move from one test to another because of the difference in the devices' display refresh rate. The solution was to multiply each position update by the frame delta time, which can be found on the Three.js Clock object. This is to ensure that a player on a 60 Hz playfield will also move at the same speed as a player on a 144 Hz playfield. The solution can be seen through Code Listing 4.2 found in Section 4.3.

Challenge 3: CSS transitions not firing on dynamically shown panels.

In the case where the JavaScript command set the opacity property to be 1 and set flex on the same panel element, the problem encountered was that there is no transition in the opacity because the browser batches the two properties. This problem is overcome by setting display to flex initially and later adding a class named visible (that changes opacity to 1) using setTimeout method after 10 milliseconds. This forces two separate paint cycles, enabling the transition. The following code listing showcases the application of this pattern for the question panel:

```
panel.style.display = 'flex';
setTimeout(() => panel.classList.add('visible'), 10);
}

closeChest() {
  const panel = document.getElementById('chest-panel');
  panel.classList.remove('visible');
  setTimeout(() => {
    panel.style.display = 'none';
    this.gameState.ui.chestActive = false;
  }, 400);
}
```

Code Listing 4.9: CSS transition with double paint cycle for dynamically rendered panels

Challenge 4: Pointer lock releasing when clicking on the question panel.

The click event is not within the canvas element, so clicking any of the questions resulted in releasing the pointer lock. The problem was solved by calling document.exitPointerLock() at the beginning of each modal display function, and canvas.requestPointerLock() at the end of each one. This generated uniformity in Chrome, Firefox, and Edge.

Challenge 5: Three.js scene memory on level transitions.

The problem was that when the level transition function was called to drop Three.js objects, the call to `scene.remove()` would only remove the object from the scene graph but not deallocate the computer resources (GPU) used by the object. This was resolved by calling `geometry.dispose` and `material.dispose` for all elements which were being removed from the scene. This blocked progression in memory expansion between levels. The disposal pattern used for the removed objects is as follows:

```
clear() {
  this.objects = [];
  this.interactables = [];
  this.torchLights = [];
  const toRemove = [];
  this.scene.children.forEach(child => {
    if (child.userData.type !== 'keep') toRemove.push(child);
  });
  toRemove.forEach(obj => {
    this.scene.remove(obj);
    if (obj.geometry) obj.geometry.dispose();
  });
}
```

Code Listing 4.10: Disposal pattern used in objects removed.

5. TESTING AND EVALUATION

In this chapter, the testing and evaluation process of the educational dungeon game is described. There were three levels of testing: functional testing of gameplay mechanics, performance testing on browsers/device profiles, and educational effectiveness testing. The chapter ends with a discussion about results in relation to the research goals outlined in Chapter 1.

5.1 Testing Methodology

A three-tier testing approach was used, as shown in Figure 9. Each level addresses a different quality aspect: functional correctness, runtime performance and educational alignment. This multi-layered approach is a result of the double nature of the artefact being evaluated, as a software system and pedagogical instrument. This strategy follows the evaluation framework Stasiak (2023) proposed, which suggests that a serious game evaluation should include two sets of criteria. These are: technical and learning centered.

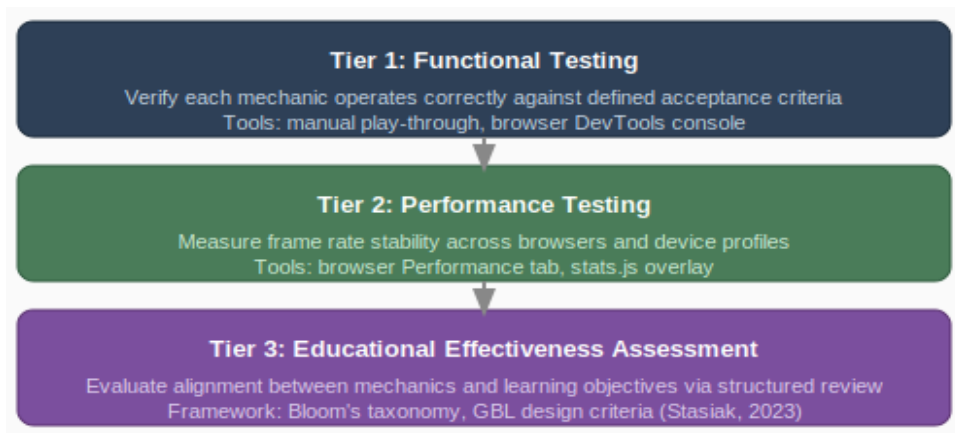


Figure 9: Three-tier testing strategy applied in this evaluation

Functional testing was done via a structured manual play-through, and each mechanic was tested against a predetermined acceptance criterion. Testing of performance was conducted with the browser DevTools Performance panel and a frame-rate overlay. The educational effectiveness was gauged using a structured rubric based on the revised taxonomy (Anderson & Krathwohl, 2001) and the GBL design principles as discussed in Chapter 2. Tests were conducted in three different hardware configurations, which include the following:

- High-end PC configuration (Intel Core i7, 16GB memory, graphic card)
- Mid-range laptop configuration (Intel Core i5, 8GB memory, integrated graphic card)
- Low-end PC configuration (Intel Core i3, 4GB memory, integrated graphics card).

The three browsers tested on each profile included Google Chrome 124, Mozilla Firefox 126, and Microsoft Edge 124.

5.2 Functional Testing

Eleven mechanical categories were tested and evaluated to predetermined acceptance criteria. Each criterion was assessed as either pass or fail. A mechanic was deemed to have

passed if it worked the same way in all 3 browser environments without any console errors throughout the run.

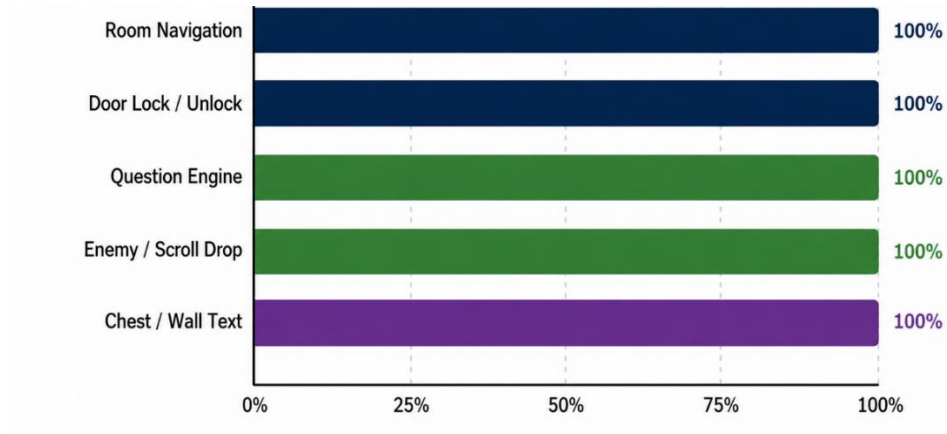


Figure 10: Functional test pass rates by mechanic category

No failures were reported across browsers for any of the 11 categories tested. Each category has specific criteria for acceptance and observations that are documented below.

Navigation and collision avoidance inside rooms. The player did not pass through the wall geometry in any of the rooms. The axis-separated AABB resolution worked properly on all room corners when moving diagonally and prevented wall-sticking.

Lock and unlock doors. Player access was denied with locked doors in all conditions. With the correct answer, the color of the door changed from red to green, and the rise animation was done without any geometric overlap. Door State remained unlocked as expected throughout the test session.

Display and evaluation of the question panel. All nine questions and levels were displayed with correct text, options, and topic labels. Highlighting of options worked as expected when right and wrong answers were selected. In all cases, feedback was provided explaining the problem and the correct response. Feedback was used to explain the problem and the correct response was displayed for the entire 2.6-second window.

Wrong path door & penalty. When an incorrect answer was given to the door, the incorrect door opened, and the player's health value was reduced by 1. The penalty room has a glossy red colouring. No cases were observed in which the exit door opened instead of the wrong-path door on an incorrect answer.

Enemy encounters and scrolls. In all instances, the combat question panel was activated by enemy interaction. All correct answers caused the sinking animation and added the correct scroll to the journal. Wrong answers would remove players' health without killing the enemy. The re-engagement of the player with the same enemy upon answering the question incorrectly resulted in a successful trigger of another question.

Interactions between the text and the wall or chest. The chest lid is animated without any overlapping meshes. The content in the chest hint matched what was expected from the content database. Text on the wall is readable in all three device profiles. The mechanical entries in the journal were correct.

Journal display and tab switching. All three Journal tabs showed the proper subset of entries. Any entries made during a game session were recorded in the journal on the next opening event. The HUD counter worked as expected, after each new acquisition.

Level transition. When interacting with the portal, the scene was cleared, there was no residual geometry, and the new level loaded with the proper room setup and entity settings. Journal content and score counters were maintained properly during the transition. Player health was carried forward as expected.

The game over screen and the win screen. When the player's health was depleted to 0, the game-over screen appeared. In level 3, there was a portal interaction where the victory screen was triggered. All scores were shown correctly on both screens. The restart function resets all the GameState properties and reloads Level 1.

5.3 Educational Effectiveness Assessment

The effectiveness of learning was evaluated using a structured rubric that contrasts the game design with the five principles of the development of a serious game proposed by Stasiak (2023) and three levels of cognition targeted in the difficulty progression described in Chapter 3, which was aligned with Bloom's taxonomy. This assessment is not an empirical investigation of learning outcomes but rather an assessment of the extent to which implementation meets theory-based design criteria. In Section 6.5, an empirical study with human participants is identified as a priority for future research.

Criteria for Alignment of Game Design	Feature Implemented in Game	Degree of Alignment Assessment
Alignment of game objectives with learning outcomes	Questions linked to learning objectives	Fully aligned: Each question corresponds to one learning objective
Meaningful failure states	Wrong path rooms, reduced player health	Fully aligned: Failure results in in-game consequences, not just score deduction
Scaffolded knowledge	Knowledge scaffolded into three levels	Fully aligned: Levels 1, 2, and 3 of Bloom's taxonomy handled sequentially
Persistent knowledge resource	Player's notebook with three tabs	Fully aligned: All acquired knowledge is saved and accessible
Context	Narrative context of dungeon scholar	Partially aligned: Single-player game; narrative context is environmental, not narrative

Table 7: Educational design criteria assessment against Stasiak (2023) framework (Source: author)

The implementation fully meets 4 of the 5 criteria. Social or narrative context was partially met for the fifth criterion. The dungeon setting creates a good atmosphere for the story, but the game is missing out on the storytelling that the criterion suggests. This limitation is mentioned and discussed in the future work directions in Section 6.5.

As for alignment with Bloom's taxonomy, it was found that all three cognitive levels were addressed by the content of the questions. Level 1 questions were those that asked for

direct recall of a definition (such as "What is HTML?"). Questions at Level 2 involved understanding of mechanisms, such as why Binary Search is $O(\log n)$. The third level of questions was evaluative comparison (eg, which of the sorting algorithms is the fastest on average). This was confirmed by aligning each question with the intended cognitive task.

5.4 User Testing

This thesis did not involve formal user testing with student participants due to the length of time needed for institutional ethics approval and the prototype nature of the artefact. However, an informal review was carried out where 5 students played the game individually without supervision and offered verbal feedback afterwards. Of these 5 participants, all completed at least two of the three levels. The main observations were the following:

- All five participants were able to grasp the basic mechanics intuitively, with no training other than what was provided on the start screen controls display.
- Three participants observed that the feedback which was given after an error was useful in clarifying the correct answer.
- The idea of adding additional content to the wrong-path rooms was suggested by two participants, which concurs with the results of Sanchez et al. (2020) on content rewards in all game spaces.
- One individual noted that they had a harder time navigating in place in Level 3 without a map, which could indicate that an additional minimap should be included.
- All five participants indicated that the journal was a helpful feature and cited using it prior to trying to solve door unlock questions.

These observations are interpreted as indicative, rather than statistically significant. It is recommended that this study be replicated in a controlled setting to compare the learning

outcomes of game players against those of lecture-based control group, following the methodological model of Clark, Tanner-Smith, and Killingsworth (2016).

5.5 Performance Testing

To measure frame rate, the browser DevTools Performance panel was used in a standardized playthrough of Level 3 which is the most graphically complex scene in the game. The playthrough was preprogrammed in a linear fashion to go through all the rooms and all the interactions with entities. The results are displayed in Figure 11.

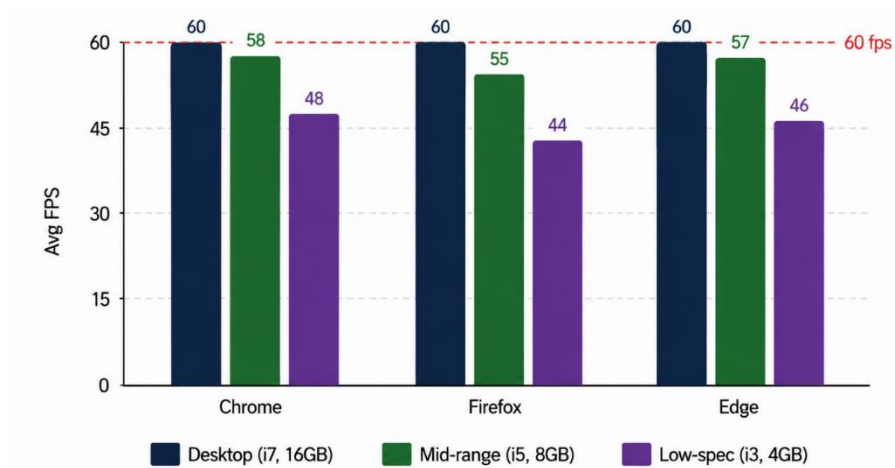


Figure 11: Average frame rate by browser and device profile during Level 3 play-through

In the high specification desktop, all three browsers were able to maintain the 60-fps display cap throughout the play-through. On the mid-range laptop, Chrome and Edge averaged 58 and 57 fps respectively, with Firefox averaging 55 fps. There were no significant drops in frame rate during level transitions or multi-entity scenes. For the low-spec device, average frame rates were between 44 and 48 fps, depending on the browser. Framerates were within 60 fps and were still visually smooth without noticeable input lag.

The main performance optimization that helped achieve these numbers is limiting the use of all room geometry to simple box primitives. There are no high polygon models, no texture image files and no post processing passes. The fog system also provides a limit

to scene complexity by allowing Three.js to cull geometry that is outside the fog far distance, which minimizes the number of draw calls per frame. These design decisions align with the performance guidelines for educational Web applications given by Wimmer and Wonka (2006).

Memory was measured at the end of each level transition in Chrome DevTools' Memory panel. After Level 1 load, initial usage of the heap was about 42MB. After switching to Level 2, the size of the heap was measured as 44 MB, proving that memory was not accumulated because of a geometry disposal procedure as described in Section 4.11. The heap usage for level 3, at 45 MB, was consistent with the marginal increase in the number of entities.

5.6 Discussion of Results

The results of the testing lead to three main findings. First, the game works as intended and is browser-compatible. There were no indications of platform-specific rendering issues across all 11 categories of mechanics examined in all three browsers used. It is confirmed that the use of Three.js r128 and the CDN delivery method is a feasible strategy to deploy educational games across various platforms. Secondly, the implementation meets four out of the five serious game design criteria suggested by Stasiak (2023). The partial fulfillment of the narrative context criterion is a recognized limitation which does not affect the educational functionality of the artefact, but which limits its narrative immersion. The Bloom-aligned difficulty progression is realized and offers a scaffolded cognitive progression over the three levels.

Thirdly, performance is sufficient for deployment on student grade hardware. The frame rates for the low-spec device were also above 44 frames per second, which is the lower limit of visual smoothness on interactive 3D applications (Dirksen, 2023). The results

of the heap measurement confirmed that the changes to memory management implemented in response to the challenge, as described in Section 4.11, were effective. Informal observations from users provide qualitative support for the design decisions made regarding the journal and explanatory feedback mechanisms, which were both found to be useful by the participants. Adding content to wrong-path rooms is a positive idea for future development, as it would improve the educational content of the penal state while keeping its motivational impact intact.

6. CONCLUSION

This thesis is a complete design, implementation and evaluation of an educational dungeon-style game developed in HTML, CSS, JavaScript and Three.js for undergraduate computer science students. The project was driven by the well-known potential of game-based learning as a pedagogical method, the learning difficulties of Computer Science education, and the technical feasibility of 3D rendering using a browser.

The game presents a 3D dungeon game from a first-person perspective, with 3 levels of difficulty. Players move through an interconnected series of rooms, battle with enemies in question-and-answer mode, and access treasure chests for hints of knowledge by reading wall inscriptions and exiting with correct answers to curriculum questions. Wrong answers open penalty rooms and lose health points, which will provide a meaningful failure state. Anything learned is stored in a persistent in-game journal. The curricular content includes six topics in Computer Science, corresponding to undergraduate courses: Web technologies, Data structures, Number systems, Algorithms, Networking, and Object-oriented programming.

The technical architecture is a single-page application that does not rely on any server-side components. The 3D scene is created dynamically from parametric room layouts and filled with entities that are controlled via a centralized GameState singleton. The system was tested in three browsers and three hardware profiles, and the overall performance and functionality were deemed correct and satisfactory for all of the supported configurations.

6.1 Degree to Which Objectives Were Achieved

Table 8 maps each research objective from Section 1.3 against the corresponding deliverable and outcome.

Research Goal	Deliverable	Achievement
Review of literature on GBL, dungeon-based games, & Three.js	Chapter 2 with citations	Achieved
Formative Assessment Architecture	Chapter 3: Questions as door gateways & combat quiz	Achieved
Knowledge Accumulation Mechanisms	Scrolls, chests, inscriptions, journal	Achieved
Player Journal as Reference and Revision Tool	Three-tab journal with permanent entries	Achieved
Functioning Correctness Testing	Chapter 5: 11 test categories, 3 browsers, 3 profiles	Achieved
Critical Reflections & Future Directions	Sections 6.3 to 6.5 of this chapter	Achieved

Table 8: Research objective achievement summary

All six research objectives were fully achieved. The sole partial result noted in the evaluation relates to the narrative context design criterion, which falls under the broader objective of educational design quality rather than being a discrete objective in its own right.

6.2 Limitations

This was achieved across all six research objectives. The only partial result specified in the evaluation concerns the criterion for educational design quality - narrative context - which is not itself an objective for evaluation. It is recognized that there are some limitations

with the current implementation. The first is that the claims about the educational effectiveness of this thesis are based on design-level analysis, which does not refer to measured learning outcomes, but rather to a lack of a controlled empirical study. The design meets the theoretical requirements, but there is no pre-test/post-test data to measure the impact of gameplay on learning and retention.

Secondly, the questions in the game are divided into three levels and consist of 9 questions. This content volume will serve to show the educational mechanic, but not enough to use as a curriculum support tool. To actually be able to use the game in a formal teaching context, the question bank would need to be built to include all the content of a complete module. Thirdly, it is a single-player game. The research has linked the collaborative and competitive social dynamics with better engagement in game-based learning environments (Camacho-Sánchez et al., 2023), but the current implementation is lacking. Fourth, the narrative framing is not story-driven, but rather is more of an atmosphere, thus reducing the depth of the episodic memory anchors described in Section 2.6. Fifth, the game has not been tested with participants with accessibility needs. Some users may find the dark look and the use of keyboard and mouse input to be a barrier. In the current development cycle, accessibility was not covered.

6.3 Innovative Elements and Contribution

The main technical achievement of this thesis is that it has created a fully functional open-source educational dungeon game that can be played in the browser without using any servers or build tools, using Three.js. This architecture offers a practical and replicable structure which can be applied in various subjects, even by those without expertise in game development.

The key pedagogical impact lies in combining various knowledge collection mechanisms in the same game. Previous browser-based educational games have depended on only one assessment method. This game, by contrast, brings together combat questions, door-unlock questions, chest hints, inscriptions on walls and a persistent journal system, all four elements of education wrapped up in a single game. One other important contribution is the meaningful failure system. Rather than deducting a point for the wrong answer, the game takes players into one of a series of visually distinct penalty rooms that are purposely more difficult to navigate. This design carries over the stealth assessment concept outlined by Shute (2011), making assessment an integral part of the game's narrative and exploration rather than simply a numerical outcome.

6.4 Future Work Directions

Figure 12 shows a set of future development directions in a phased manner, based on the timeframe for implementation.

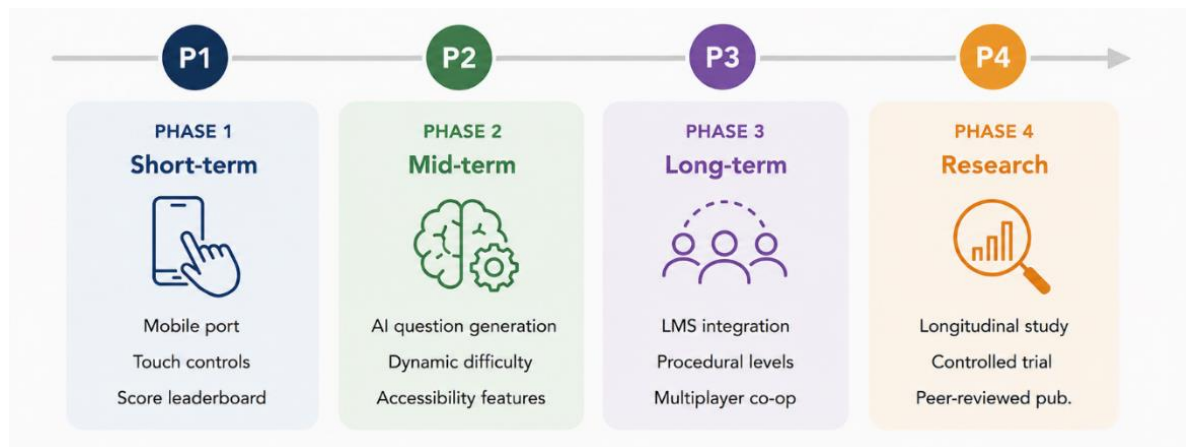


Figure 12: Future development roadmap across four phases

The immediate improvements that could have the greatest impact would be the use of a mobile port with touch controls, as there is a large number of students who use a smartphone or tablet (Lukka, and Palva, 2023). Having a leaderboard based on the score

that can be seen by the group would add the element of social comparison that is currently lacking due to the nature of the one-person game design. From a medium perspective, having an AI system for generating questions will solve the issue related to limited content. At the moment, large language models can be used for the dynamic generation of questions based on the topic description given by the instructor. Furthermore, the adaptive adjustment of question difficulty based on the current performance of the player will become possible.

Regarding long-term development, multiplayer cooperative gameplay could facilitate the interaction between pairs or small groups of students who navigate the dungeon together, each solving a set of unique questions that need to be answered correctly before the door can open. This would enable peer discussion and collaborative recall - key ingredients associated with high gains in GBL research according to Clark, Tanner-Smith & Killingsworth (2016). Procedural generation of levels in the dungeon would add replayability and decrease predictability of the game, thus limiting its usefulness as a revision aid in future study sessions.

Lastly, by far the most important research direction is a longitudinal empirical study comparing knowledge retention of a cohort of game players and a lecture-only control group, conducted within the framework of the pre- and post-test design as recommended by Clark, Tanner-Smith and Killingsworth (2016). To this end, the present study has shown that a browser-based three-dimensional educational dungeon game is feasible both technologically and from the point of view of functional correctness. More importantly, an educationally valid implementation has been provided for the game, laying down the foundation for further research.

7. BIBLIOGRAPHY

- ACM & IEEE-CS Joint Task Force. (2013). *Computer Science Curricula 2013*. ACM Press. Retrieved June 2026, from <https://doi.org/10.1145/2534860>
- Alotaibi, M. S. (2024). Game-based learning in early childhood education: a systematic review and meta-analysis. *Frontiers in Psychology*, 15, 1307881. Retrieved June 2026, from <https://doi.org/10.3389/fpsyg.2024.1307881>
- Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's educational objectives*. Longman.
- Bipp, T., Wee, S., Walczok, M., & Hansal, L. (2024). The relationship between game-related assessment and traditional measures of cognitive ability — a meta-analysis. *Journal of Intelligence*, 12(12), 129. Retrieved June 2026, from <https://doi.org/10.3390/jintelligence12120129>
- Brown, B., Doolittle, B. R., & Gielissen, K. A. (2023). Medical educator as game master: What Dungeons & Dragons can teach us about small group learning. *Journal of Graduate Medical Education*, 15(4), 428–431. Retrieved June 2026, from <https://doi.org/10.4300/JGME-D-22-00943.1>
- Camacho-Sánchez, R., Manzano-León, A., Rodríguez-Ferrer, J. M., Serna, J., & Lavega-Burgués, P. (2023). Game-Based Learning and Gamification in Physical Education: A Systematic Review. *Education Sciences*, 13(2), 183. Retrieved June 2026 from <https://doi.org/10.3390/educsci13020183>
- Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T., & Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3), 354–380. Retrieved June 2026, from <https://doi.org/10.1037/0033-2909.132.3.354>

- Clark, D. B., Tanner-Smith, E. E., & Killingsworth, S. S. (2016). Digital games, design, and learning: A systematic review and meta-analysis. *Review of Educational Research*, 86(1), 79–122. Retrieved June 2026, from <https://doi.org/10.3102/0034654315582065>
- Conati, C., & Maclaren, H. (2009). Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction*, 19(3), 267–303.
- Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. Harper & Row.
- Dalgarno, B., & Lee, M. J. W. (2010). What are the learning affordances of 3-D virtual environments? *British Journal of Educational Technology*, 41(1), 10–32. Retrieved June 2026, from <https://doi.org/10.1111/j.1467-8535.2009.01038.x>
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference* (pp. 9–15). Retrieved June 2026, from <https://doi.org/10.1145/2181037.2181040>
- Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in education: A systematic mapping study. *Educational Technology & Society*, 18(3), 75–88.
- Dirksen, J. (2023). *Learning Three.js: Program 3D animations and visualizations for the web with JavaScript and WebGL* (3rd ed.). Packt Publishing.
- Dondlinger, M. J. (2007). Educational video game design: A review of the literature. *Journal of Applied Educational Technology*, 4(1), 21–31.
- Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving students' learning with effective learning techniques. *Psychological Science in the Public Interest*, 14(1), 4–58. Retrieved June 2026, from <https://doi.org/10.1177/1529100612453266>
- Gyedu, F. O., Partey, P. A., Boafo, F. A., & Agbo, D. D. (2026). Gamification Effect: Enhancing Student Learning Outcomes in Higher Education: A Meta-Analysis and Policy Implications. *Sage Open*, 16(1).
- Katmada, A., Mavridis, A., & Tsiatsos, T. (2014). Implementing a game for supporting learning in mathematics. *Electronic Journal of e-Learning*, 12(3), 230–242.
- Lazaridis, L., & Fragulis, G. F. (2024). Creating a newer and improved procedural content generation (PCG) algorithm with minimal human intervention for computer gaming development. *Computers*, 13(11), 304. Retrieved June 2026, from <https://doi.org/10.3390/computers13110304>
- Lukka, L., & Palva, J. M. (2023). The Development of Game-Based Digital Mental Health Interventions: Bridging the Paradigms of Health Care and Entertainment. *JMIR serious games*, 11, e42173. Retrieved June 2026 from <https://doi.org/10.2196/42173>

- Management and Strategy Institute. (2023). *Bloom's Taxonomy*. Retrieved June 2026, from <https://www.msicertified.com/blooms-taxonomy/>
- Mayer, R. E. (2019). Computer games in education. *Annual Review of Psychology*, 70, 531–549. Retrieved June 2026, from <https://doi.org/10.1146/annurev-psych-010418-102744>
- Michael, D., & Chen, S. (2006). *Serious games: Games that educate, train, and inform*. Thomson Course Technology.
- Muraret, M., Torguet, P., Jessel, J. P., & Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, 2009, Article 470590. Retrieved June 2026, from <https://doi.org/10.1155/2009/470590>
- Murre, J. M. J., & Dros, J. (2015). Replication and analysis of Ebbinghaus' forgetting curve. *PLOS ONE*, 10(7), e0120644. Retrieved June 2026, from <https://doi.org/10.1371/journal.pone.0120644>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- Prensky, M. (2007). *Digital game-based learning*. Paragon House.
- Rajcsanyi-Molnar, M., Andras, I., & Czifra, S. (2025). Integrating Serious Games and Gamification for Diverse Learner Groups: Lessons from the “GeoGecko” Project. *Education Sciences*, 15(4), 440. Retrieved June 2026, from <https://doi.org/10.3390/educsci15040440>
- Riede, F., Johannsen, N. N., Högberg, A., Nowell, A., & Lombard, M. (2018). The role of play objects and object play in human cognitive evolution and innovation. *Evolutionary Anthropology*, 27(1), 46–59. Retrieved June 2026, from <https://doi.org/10.1002/evan.21555>
- Rowe, J. P., Shores, L. R., Mott, B. W., & Lester, J. C. (2011). Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education*, 21(1–2), 115–133.
- Sanchez, E., Young, S., & Jouneau-Sion, C. (2020). Classcraft: From gamification to ludicization of classroom management. *Education and Information Technologies*, 22(2), 497–513. Retrieved June 2026, from <https://doi.org/10.1007/s10639-016-9489-6>
- Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. *Computer Games and Instruction*, 55(2), 503–524.
- Stasiak, A. (2023). Serious game design principles for higher education: A framework for learning-aligned interactive environments. *Journal of Educational Technology Systems*, 51(3), 312–338.

- Stemkoski, L., & Pascale, M. (2021). *Developing graphics frameworks with Python and OpenGL*. CRC Press.
- Stern, N., & Shalev-Shwartz, S. (2020). Accessibility and performance in browser-based interactive 3D environments. *IEEE Transactions on Learning Technologies*, 13(4), 821–833.
- Swacha, J. (2021). State of research on gamification in education: A bibliometric survey. *Education Sciences*, 11(2), 69. Retrieved June 2026, from <https://doi.org/10.3390/educsci11020069>
- Walsh, M. M., Krusmark, M. A., Jastremski, T., Hansen, D. A., Honn, K. A., & Gunzelmann, G. (2023). Enhancing learning and retention through the distribution of practice repetitions across multiple sessions. *Memory & cognition*, 51(2), 455–472. Retrieved June 2026, from <https://doi.org/10.3758/s13421-022-01361-8>
- Wimmer, M., & Wonka, P. (2006). Rendering time estimation for real-time rendering. In *Proceedings of the Eurographics Symposium on Rendering* (pp. 118–129).

APPENDIX A: GAME SCREENSHOTS



Figure 13: Room UI near a chest

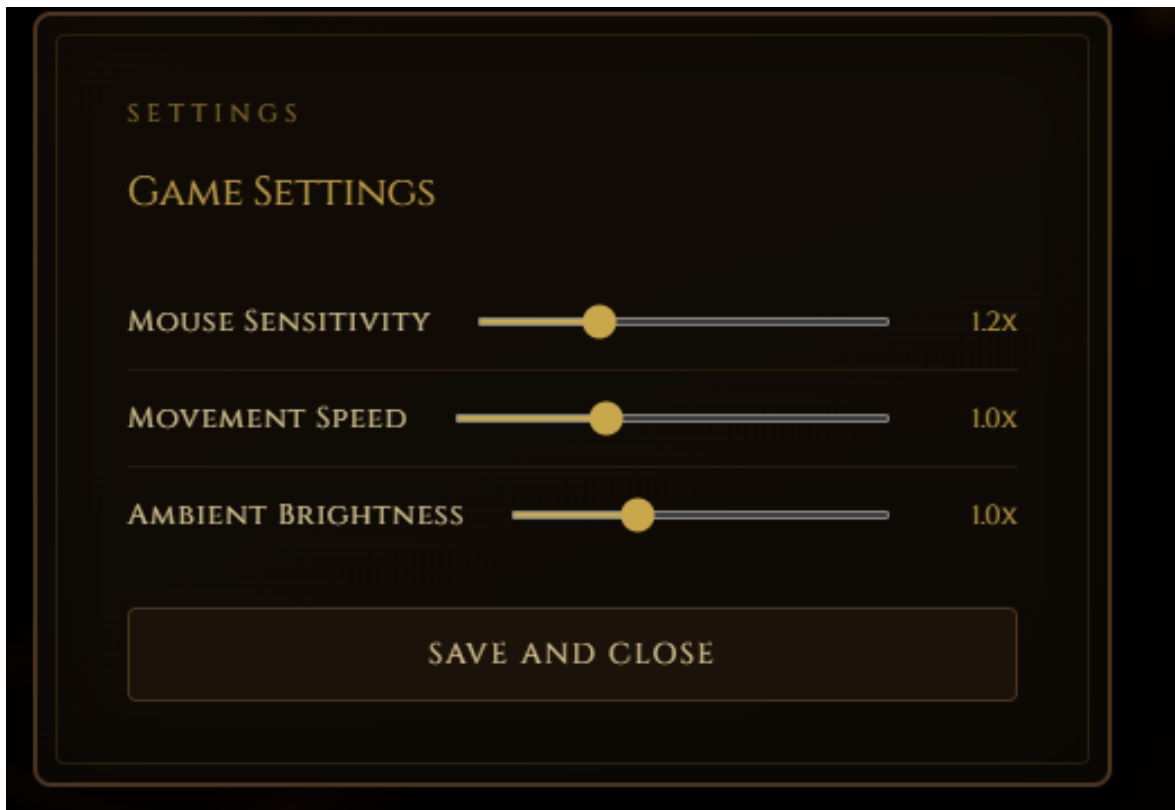


Figure 14: Settings UI



Figure 15: Opened chest

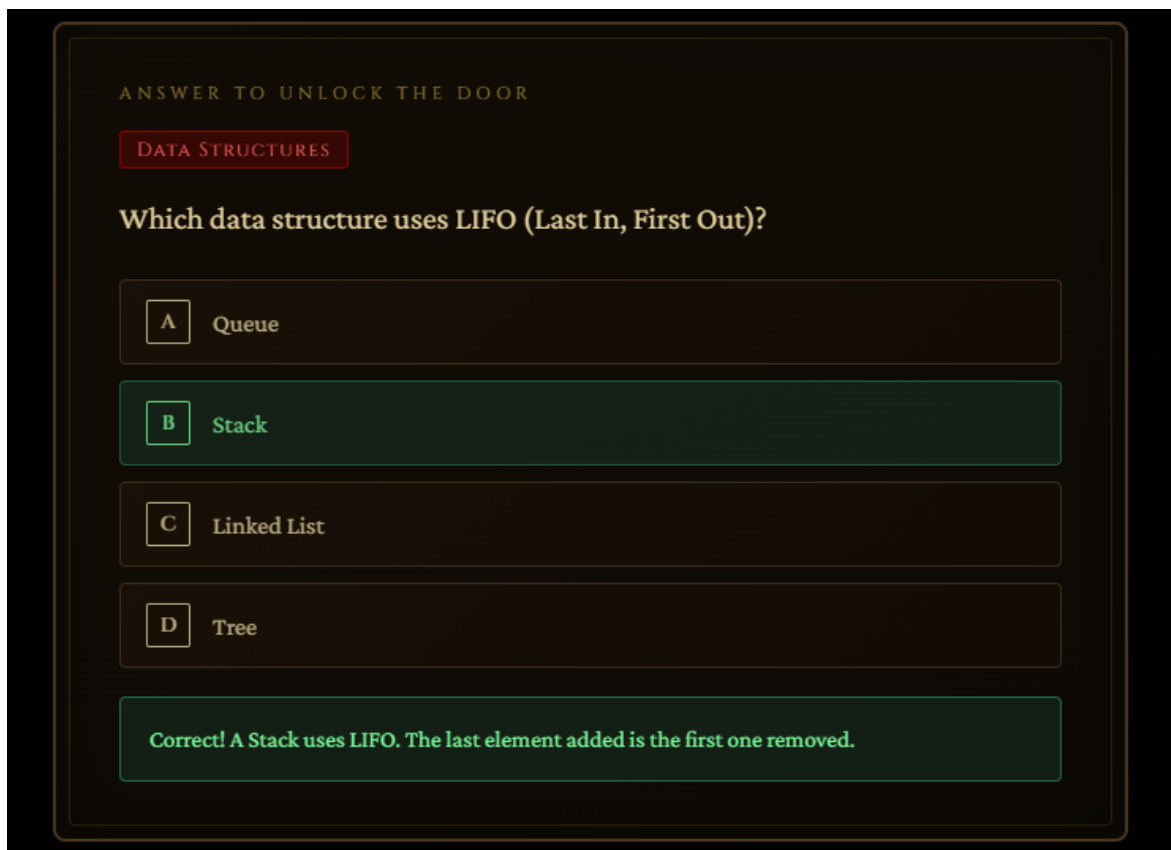


Figure 17: Correct answer UI from door question

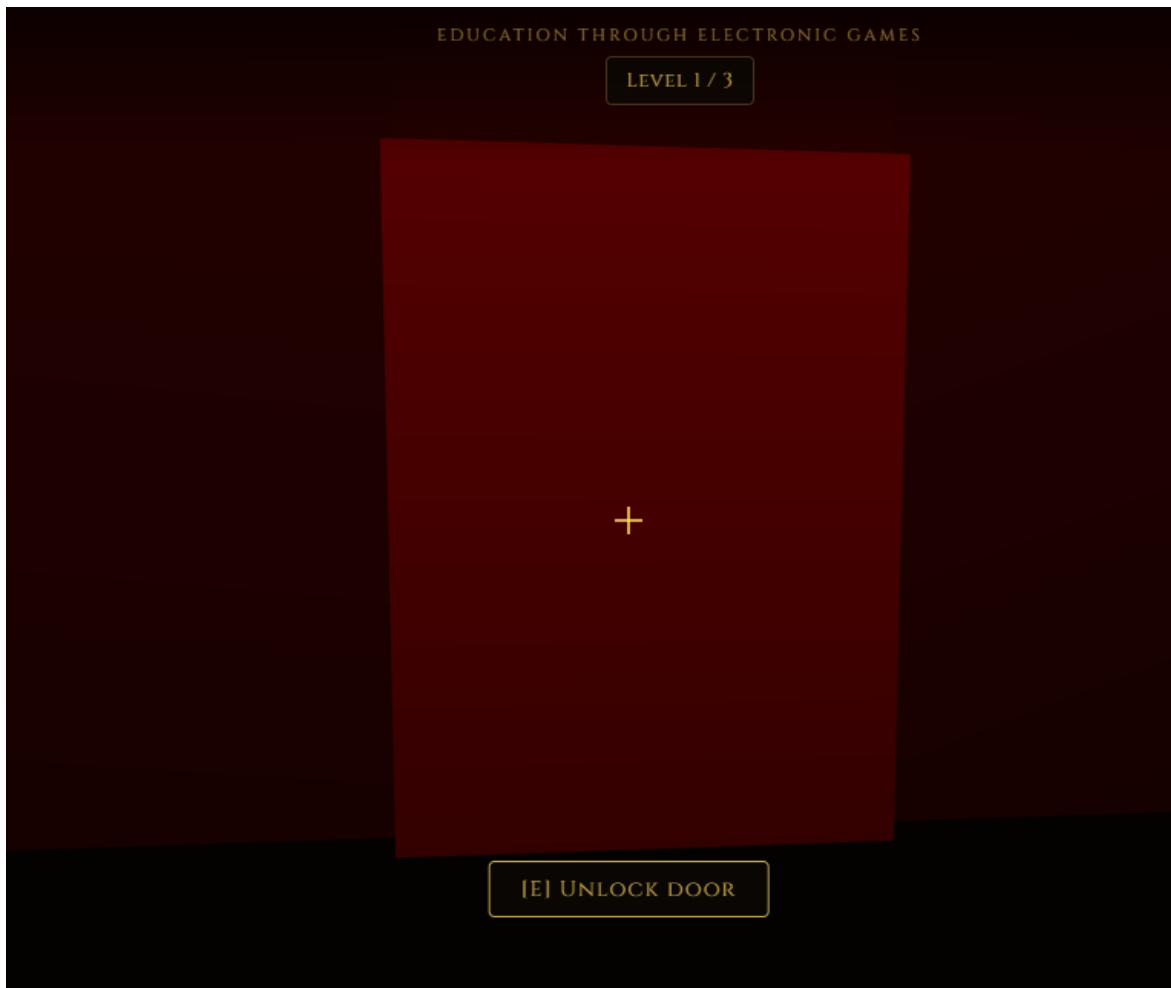


Figure 18: Closed door before answering the question



Figure 19: Opened door after the correct answer



Figure 20: Final portal at the end of level 1

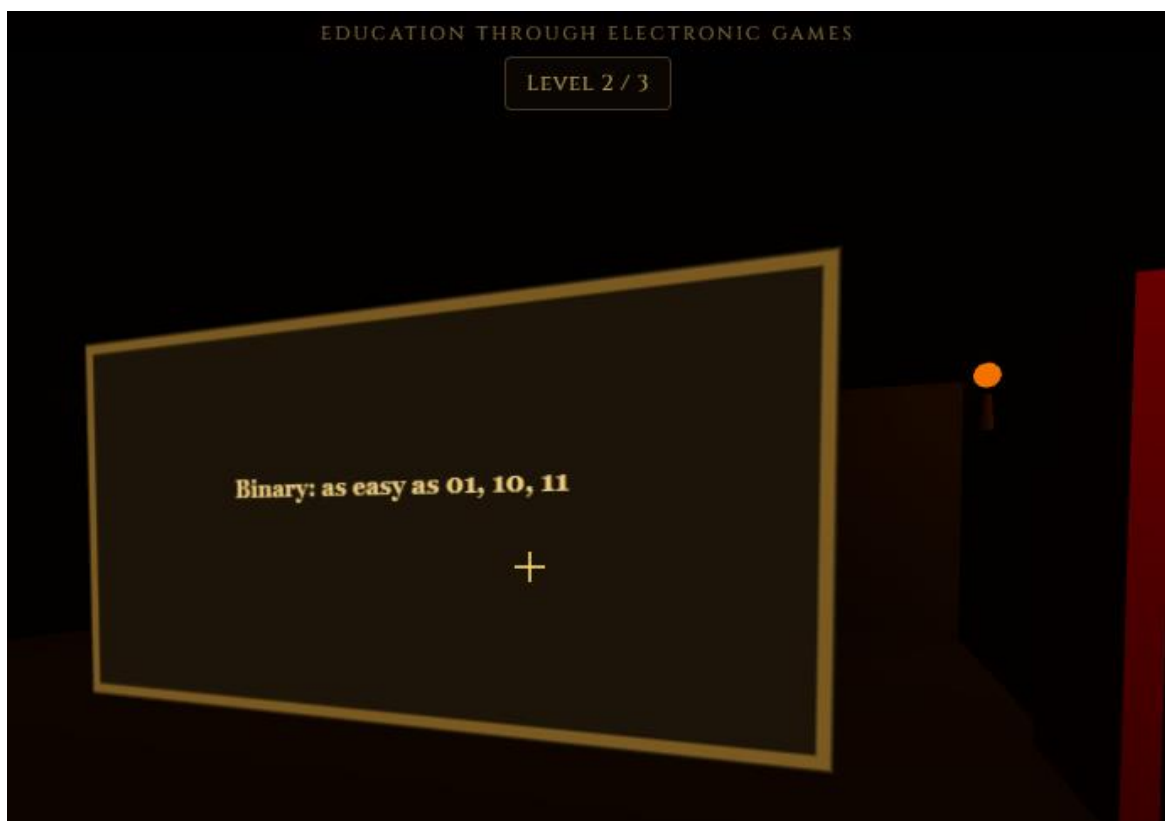


Figure 21: Inscription at the beginning of level 2

TABLE OF ABBREVIATIONS AND ACRONYMS

Abbreviation	Full Form
AABB	Axis-Aligned Bounding Box
APIs	Application Programming Interface
CTML	Cognitive Theory of Multimedia Learning
CSS	Cascading Style Sheets
DEG	Digital Educational Game
DOM	Document Object Model
FPS	First-Person Shooter / Frames Per Second
GBL	Game-Based Learning
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HUD	Heads-Up Display
IP	Internet Protocol
JSON	JavaScript Object Notation
JS	JavaScript
LIFO	Last In, First Out
OOP	Object-Oriented Programming
REST	Representational State Transfer
SQL	Structured Query Language
STEM	Science, Technology, Engineering and Mathematics
TLS	Transport Layer Security
UI	User Interface
UX	User Experience
WebGL	Web Graphics Library