University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

SortQuest: Game-based learning for sorting algorithms

Euan Halliday

March 22, 2024

Abstract

Algorithmic concepts form a crucial foundation in the computer science curriculum, yet traditional teaching methods often struggle to engage students effectively. Informed by existing game-based learning theories and tools, the design and development of SortQuest is executed. SortQuest is a computer game designed to support students in their learning of basic sorting algorithms such as bubble sort, quick sort, and merge sort. Unlike traditional teaching methods, SortQuest makes use of different interactive game modes, a 2D platformer world, pseudocode challenges, visualization challenges, and many more gamification features to teach and engage students about sorting algorithms effectively. By evaluating the game, it could be shown that players become more confident in sorting algorithms and that the gamification features integrated into the learning environment are a great success.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Euan Halliday Date: 22 March 2024

Contents

1 Introduction

1.1 Motivation

Learning how algorithms work is an important milestone for computer scientists on their journey towards becoming proficient coders. However, classes that teach algorithms often lean heavily towards theory, creating a significant hurdle for many students in grasping and engaging with these critical concepts. Traditional teaching methods, such as slide presentations, visualization tracing, and textbooks, tend to become repetitive and cumbersome, potentially hindering the learning experience (Boticki et al. 2012).

This project is motivated by the potential of game-based learning and recognizes the need for a more engaging and effective educational activities and tools in teaching algorithms. This approach involves teaching specific concepts through gameplay and integrating elements commonly found in popular games into a dedicated learning environment. Extensive research underscores the effectiveness of a game-based approach, increasing engagement and fundamentally transforming the learning process into an enjoyable and rewarding experience. Games offer students a break from reality, allowing them to immerse themselves fully in the worlds these educational games provide.

As students delve into the complex nature of learning algorithms, the initial overwhelm induced by heavy theoretical content can be alleviated within a more relaxed and gaming-oriented setting. The belief that game-based learning offers a more exciting, enjoyable, and engaging way of learning inspires this project. Therefore, the project motivation is to create an educational environment, by devising a game-based learning tool, where learning sorting algorithms is not just informative but also an enriching and pleasurable experience for students.

1.2 Aims

The primary aim of this project is to create a game-based learning tool that teaches sorting algorithms to students learning computing science. The approach to creating the game will involve teaching the student easier-to-grasp sorting algorithms, such as bubble sort, and then moving on to more challenging sorting algorithms, specifically quick sort and merge sort. From this overall goal, we can create sub-goals for the game such as:

- **Interactive Learning Modes:** Teaching sorting algorithms to students through an array of interactive game modes which integrate a dynamic and engaging learning environment.
- **Gamification Integration:** Conducting background research into game-based learning tools and gamification features, therefore allowing the successful integration of the findings from this research into the game.
- **Engaging Educational Experience:** Creating a game that surpasses the limitations of traditional teaching methods, ensuring an enjoyable and captivating experience for the learners.
- **Comprehensive Evaluation:** Evaluating the system's usability, teaching effectiveness, gamification elements, and overall user experience through an evaluation involving computing students.

2 Background

2.1 Game-Based Learning and Gamification

Game-based learning can be described as an educational approach that uses game-play principles, often digital but not exclusively, to achieve specific learning objectives (Jan L. Plass and Kinzer 2015). This approach emphasizes the use of games as a central component in the educational process. The design of these games is aimed at achieving specific learning outcomes by balancing them with enjoyment and engagement of gameplay. Therefore, game-based learning transforms educational content into interactive and captivating experiences while providing a more exciting way for learners to acquire knowledge and skills.

Gamification involves adding game elements into non-game contexts to enhance engagement and interactivity. This strategy is particularly effective for the learning preferences of millennials and post-millennials who thrive in adaptable learning environments (Begosso et al. 2018). Gamification allows learning objectives to be achieved through the features that the game offers. These features can come in the form of incentive systems such as points, badges, levels, rankings, streaks etc. and can elevate the overall enjoyment of learning. For example, Duolingo, a game-based learning tool, uses nearly all the features mentioned above to create maximum engagement and has become one of the most successful game-based learning tools in the modern era (Duolingo 2019). These features can motivate learners and create an increased retention of educational content. This aligns with the definition of a game as "a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome" (Jan L. Plass and Kinzer 2015).

Furthermore, gamification can provide a social dimension by incorporating features like competition through leaderboards, collaboration through multiplayer challenges and feedback from other users, creating a more engaging and motivating environment. Gamification can be seen as most effective when the "sweet spot" is hit, which can be described as players succeeding but with some struggle, indicating a state of 'flow' (Jan L. Plass and Kinzer 2015). This supports continuous learning by providing ongoing challenges and opportunities for improvement and indicates a continuous feedback loop that fosters a lifelong learning mindset.

2.2 The Theory Of Game-Based Learning

This section aims to provide an overview of the theory behind game-based learning by identifying key game-based learning behaviours such as interest, cognitive engagement, motivation and immersion that can all be utilized within the game-based learning tool.

2.2.1 Interest

In this subsection, we delve into the subject of **interest** as discussed in the paper (Hidi and Renninger 2006). Understanding the dynamics of interest development is crucial in the context of game-based learning, and the four-phase model which provides a comprehensive framework for understanding interest development:

(Phase 1) Triggered situational interest: occurs when learners are introduced to a new concept or activity, in this case, a game-based learning tool, but it can also be sparked by learning environments that include puzzles, computers, personal relevance or surprising information.

(Phase 2) Maintained Situational interest: occurs if the activity continues to engage the learner, whose interest shifts from a temporary phase to a more sustained one. This means the learner will spend extended time focused on the present activity however, re-engagement of the particular activity over time is not guaranteed.

(Phase 3) Emerging Individual Interest: highlights the initial inclination to re-engage with an activity due to the learner gaining positive feelings, stored knowledge and curiosity questions, which encourages re-engagement. In this phase, learners become more influenced to engage in the activity due to a self-generated interest as opposed to an externally generated interest in the previous phases.

(Phase 4) Well-Developed Individual Interest: occurs when a learner has a lasting inclination to continuously revisit and engage with the activity over an extended period. The learner continues to gain positive feelings, extensive knowledge, and the ability to control their learning processes independently. In this phase, the learner is completely interested through self-generation, while external support is occasionally given as a precaution in order to encourage the learner to remain in this phase.

This model can be applied in the context of game-based learning. In the early phases, educators can leverage the model to sustain the learner's attention by introducing exciting gamification elements. Setting goals in the game aligns with the emerging individual interest phase and can encourage learners to take accountability for their learning journey. In addition, providing positive feelings and support during the triggered and maintained situational interest phases is crucial because it allows an environment for interest to evolve naturally.

The shift from external to internal support can also be pivotal for developing interest in a game. The game should transition from providing support and encouraging students to learn, to the student learning from their own developed interest. Furthermore, there is a strong correlation between interest development and students' feelings of self-efficacy. In the context of game-based learning focused on sorting algorithms, as the student's self-efficacy grows in sorting algorithms, so does their interest (Hidi and Renninger 2006).

2.2.2 Cognitive Engagement And Motivation

Games designed for entertainment have a proven track record of keeping learners motivated over extended periods (Jan L. Plass and Kinzer 2015). This motivation is driven by incorporating features that engage learners, forming the foundation of effective game-based learning. For game-based learning to be truly effective, it must achieve cognitive engagement, as activating the cognitive processes is the key to a meaningful learning experience (Jan L. Plass and Kinzer 2015). Game-based learning emphasises the construction of mental models by learners. This process involves the "selection, organization, and integration" (Jan L. Plass and Kinzer 2015) of the information presented in the game and blending it with the learner's knowledge. Developers also play a key role in shaping the learner's journey by strategically managing cognitive load during game-play. This involves "Minimizing extraneous processing, skillfully handling essential processing, and encouraging generative processing"(Jan L. Plass and Kinzer 2015), which aims to create effective cognitive engagement. Generally, from a motivational standpoint, game-based learning centres on engaging and motivating players to elevate their learning experiences, propelling players into cognitive processing of game content and translating this cognitive processing into improved learning goals.

2.2.3 Immersion

In this subsection, the subject of **immersion** is discussed, with reference to the paper (Shi and Shih 2015). Immersion can be referred to as the extent to which individuals become engaged and deeply involved in a particular activity or environment. When applying immersion to game-based learning, it can be described as the extent to which players are absorbed in the game environment, shifting their awareness from the external world to the current virtual setting. It is important because a higher level of immersion causes a stronger motivation among users to accomplish the goals of the game.

The fantasy of a game, which includes the narrative and sensory sensation, plays a crucial role in providing immersion. Compelling storylines and realistic sensory experiences contribute to a sense of presence in the game and are vital for maintaining a player's attention. Realistic sensory stimulation can be described as the audiovisual elements that stimulate the senses and create a more immersive experience.

Furthermore, **challenges embedded within a game** contribute to immersion by requiring active participation and problem-solving. This is particularly effective in educational games as these challenges can be aligned with the game's learning objectives. However, the design of these challenges must create a balance. Excessively difficult challenges may lead to player frustration and disinterest, while overly simplistic challenges can induce boredom. Thus, the emphasis lies on maintaining a balance which activates a state of 'flow'.

Interactivity is also fundamental to creating immersion and can be done by establishing a dynamic relationship between the player and the game system. The game must have meaningful interactions where the player's actions have consequences. This can be done by including responsive feedback to a player's action in a game, reinforcing a sense of agency by making learners realise that their decisions in the virtual environment will have consequences.

Another factor that contributes to immersion is **freedom and exploration**. Allowing learners to navigate the game at their own pace and explore the learning environment freely contributes to the immersive quality of a learning environment. Learners, therefore, become empowered as they can navigate the learning environment and make choices that align with their preferences.

Incorporating a **social or collaborative dimension** into the game further enhances immersion through competition, cooperation and communication, creating a more dynamic and engaging environment. Whether through friendly competition or collaborative problem-solving, introducing a social aspect increases the overall sense of presence as players interact with real individuals within the virtual environment.

All the above elements are essential in creating a more dynamic and engaging environment, which enhances motivation and, importantly, a profound sense of immersion. They will be incorporated into the development of the game-based learning tool.

2.2.4 Models of Learning

Research models of learning serve as a justification for understanding the underlying principles driving effective game-based learning. By examining instructivist and constructivist models, we aim to embed strategies prioritising learner engagement and interaction with the educational content.

The **instructivist** learning model can be described as the acquisition of knowledge facilitated by a teacher or instructor (Barr 2019). This approach is deeply ingrained in traditional educational settings and involves transmitting information from an authoritative figure to learners. Regarding game-based learning, the instructivist model can be seen in tutorials that introduce the player to the game mechanics and goals. However, game-based learning tools often try to avoid the

instructivist approach as it can negatively impact immersive experiences, and players would rather learn through exploring the game environment.

The **constructivist** model of learning can be described as an "active process through which learners may themselves construct new knowledge, by applying existing knowledge to new problems" (Barr 2019). It is a learner-centric approach that highlights the idea that knowledge is personally constructed based on experiences of completing tasks and thinking independently. (von Glasersfeld 1995) explain that constructivism involves learning general ideas that become the starting point for recognizing and solving specific problems. Therefore, it can be seen as a more individual approach to learning where the player has to construct the meaning of what they are being taught. In game-based learning this can be related to the game environment serving as a context for drawing on existing understanding. For example, many well-designed games minimize explicit instructions to motivate the player to construct their own understanding.

The examination of instructivist and constructivist models underscores the importance of prioritizing learner engagement, and choosing how the interaction between the learner and educational content occurs, tailoring strategies to individual needs, and designing effective game-based learning experiences.

2.2.5 Design Models Of Game-Based Learning

Given the descriptions of the learning models above, both design models of game-based learning, Fig. 2.1 and Fig. 2.2, align more closely with constructivist learning principles rather than instructivist principles. This is because they emphasize learner engagement, immersion and active participation in the learning process, which are central to the constructivist learning theory.

One design model of games is the challenge, response, feedback model (Jan L. Plass and Kinzer 2015). This model is prevalent in most games where the player is presented a challenge, which causes a response from them, and this response determines the type of feedback they receive from the game. The loop in learning then occurs when the feedback prompts a new challenge or another response. Fig. 2.1 shows how the game design characteristics are at the centre and how these elements are picked to define the learning experience.

The design model of games in Fig. 2.2 below is more concerned with the creation of games and making them immersive (Shi and Shih 2015), inspiring engagement and involvement in the virtual world as described above. The model begins with defining the game goals which are closely linked to the learning objectives. Therefore, firstly, the learning objectives of the game must be clear. Next, the fantasy defines the scenery of the virtual world, captivating players during game-play. The model incorporates the challenge factor which is also crucial in game-based learning, where challenges are generated from learning objectives and test a player's knowledge and skills. However, the level of challenge needs to be adjusted based on the player's knowledge to avoid any negative impacts, such as frustration. Narratives are important as they encapsulate learning content and challenges, making them more familiar and engaging. Furthermore, game mechanisms should be thoroughly planned as they determine the interactions between the player and the system. Therefore, combining game mechanisms with the storyline enhances player absorption. Additionally, teaching materials should be integrated into the narrative and fantasy rather than just being added straight from the slides to create a more immersive and engaging experience. The model also includes sensation described as the sensory experiences provided to a player whilst playing the game. The following attributes within the model are considered important for varied reasons: Freedom extends a player's choices in a game and is important for increasing the level of difficulty, mystery is important because it inspires players to explore and learn more about the game whilst maintaining their interest and engagement, game value can be considered from the perceived benefits, enjoyment and meaningful experiences that players derive from engaging in an educational game and sociality is important if the game is to include co-operative learning and can be described as the communication interfaces of a game. Finally,

feedback is essential in educational games as it guides students through their learning process.

Therefore, the game-based learning features chosen include those that prioritize the learner's engagement, and immersion and promote active participation. This decision aligns with the constructivist learning principles, as it encourages learners to construct their own understanding through hands-on exploration and interaction with the educational outcome.

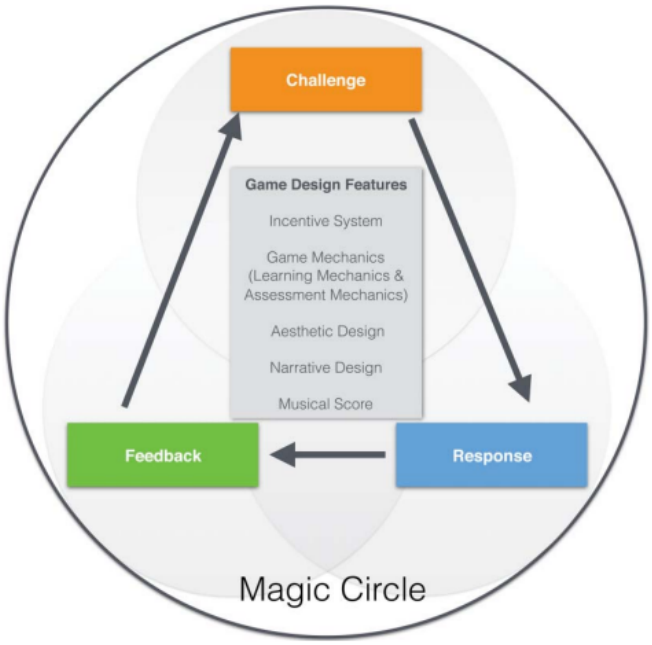


Figure 2.1: *Challenge, Response, Feedback Model (Jan L. Plass and Kinzer 2015)*

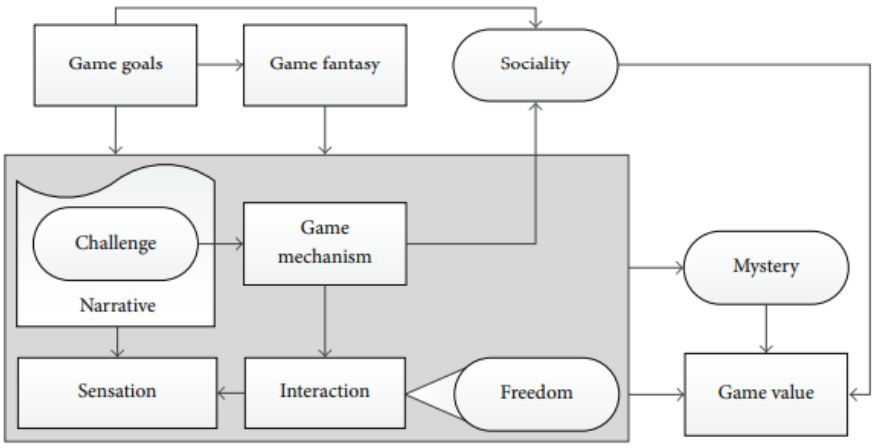


Figure 2.2: *Game-Based Learning Design Model (Shi and Shih 2015)*

2.3 Existing Game-Based Learning Apps

Researching existing game-based learning tools can provide valuable insights into effective strategies for integrating gaming elements into educational frameworks. Game-based learning tools have become ubiquitous and have helped significantly educate learners in many subjects. Their success lies in the integration of gaming elements into educational frameworks. Therefore, in this section, we will review different game-based learning tools and gather insight into their features.

2.3.1 DeCode

DeCode is an educational game designed to teach data structures and algorithms using gamification principles identified in literature (Su et al. 2021). These principles include interactivity, such as explanations, using data structure and algorithm questions, prediction exercises, creating game objects using real-world metaphors and employing a 2.5D aesthetic environment for visual appeal. It uses a car-park metaphor to represent parking spaces as memory locations, vehicles as data elements and vehicle movement as the execution of assignment commands. The game-play is structured into four levels, where each level gets progressively harder, covering the topics of arrays, ArrayLists, stacks and queues. Each level begins with a basic data structure description and allows players to explore and interact with its operations. It gradually increases difficulty within and between each level to motivate and engage learners. The game interface integrates visualizations, task descriptions, and pseudocode representations, facilitating the connection between visual and code-based learning.

Figure 2.3: *DeCode (Su et al. 2021)*

2.3.2 Sortko

Sortko is a mobile game-based learning tool which teaches students sorting algorithms such as bubble sort, shell sort, quick sort and insertion sort (Boticki et al. 2012). The game is played in horizontal mode and has three main views: the initial view, the sorting view and the results view. It also includes different modules, such as the interactive sorting client module, the scaffolding client module, and the motivational module. The interactive sorting client module verifies whether students' sorting steps align with the chosen sorting algorithm. The scaffolding client module provides textual help messages to guide students through the sorting process and correct any errors. The motivational module awards points to students based on their accuracy, speed

and the type of algorithm used. Where these points are then saved and contribute towards the student's position on the public leaderboard. However, points can also be deducted for unsuccessful actions during the sorting process.

Figure 2.4: *Sortko (Boticki et al. 2012)*

2.3.3 Duolingo

Duolingo is a well-renowned and successful game-based learning tool which teaches over 40 languages and has even moved on to teaching music and maths. It provides a gamified learning experience by using interactive exercises and quizzes, which become more difficult as a user progresses through their course. Duolingo is also well known for its user-friendly interface, adaptive learning approach, and the ability to track progress. It offers a large range of gamification features such as badges, streaks, experience points, and leaderboards to positively impact user engagement. Users perceive these gamification features as motivators to continuously reengage with the application and compete with other users. For example, users may be reengaging to keep up their streak or to maintain their position in the leaderboards.

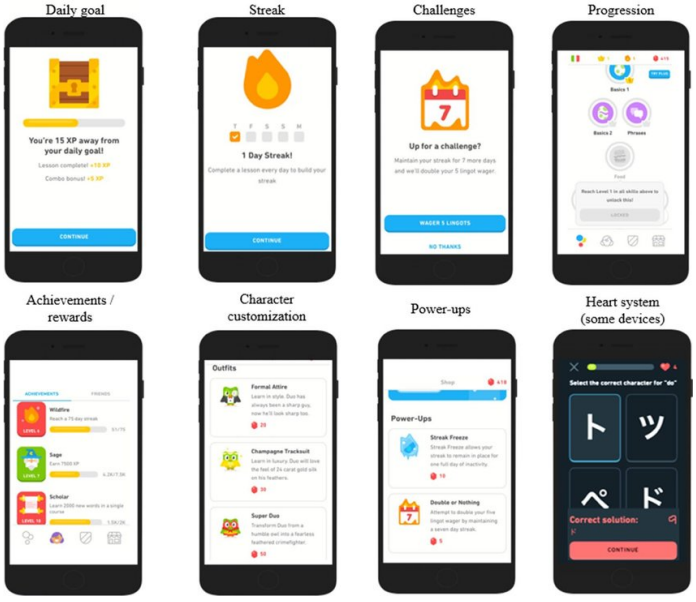


Figure 2.5: *(Mitchell Shortt and Akinkuolie 2023)*

From the analysis of Decode, Sortko, and Duolingo, several key features and considerations

emerge that inform the approach to developing a game-based learning tool. Firstly, the emphasis on interactivity and representation of real-world objects in DeCode highlights the importance of providing learners with opportunities to actively explore and interact with educational content, while using metaphors or analogies to represent computing constructs. Sortko's inclusion of feedback mechanisms and leaderboards highlights the value of providing learners with guidance and a social dimension. Furthrmore, Duolingo's success in leveraging gamification features to sustain user engagement, indicates the importance of incorporating these gamified elements into game-based learning tools.

2.4 Effectiveness of Game-Based Learning In Computing Science

Below are two studies in which a game-based learning tool was used to teach a computer concept and then evaluated. The study also examines the effectiveness of visualizations, to influence how they can be integrated into a game-based learning context.

2.4.1 Teaching Sorting Algorithms

(Lim et al. 2022), created a sorting algorithm game which includes bubble sort, quick sort, selection sort, and insertion sort, and it was created to measure the effectiveness of game-based learning. The game was created using Unity3D and includes a tutorial mode and a game mode which caters to different learning objectives. The tutorial mode aimed to facilitate self-learning through teaching slides and visualisations, while the game mode allowed students to practice sorting algorithms by swapping robots. Additionally, the game included a point system to evaluate player performance, adding a competitive and motivating element. To evaluate the game's effectiveness, ten students were split into two groups of five, Group 1 and Group 2. Group 1 involved students using traditional self-directed learning methods to learn sorting algorithms, whilst Group 2 involved students using the game-based learning tool to learn sorting algorithms. Results show that Group 2 outperforms Group 1 by completing tests more quickly and achieving higher scores, as shown in Fig. 2.6. Moreover, Group 2 also had many positive responses; students believed it improved their understanding of sorting algorithms. Therefore, this study highlights that teaching sorting algorithms using game-based learning tools is advantageous and encouraging as the scope of this project is based on teaching sorting algorithms using game-based learning tools.

(a) *Average Time taken to complete tests* ***(b)*** *Average score*

Figure 2.6: *Group 1 vs Group 2 Results (Lim et al. 2022)*

2.4.2 Teaching Constraint Satisfaction Problems

Algorithms are complex and challenging to teach because they often involve abstract concepts, mathematical notions, and dynamic data structure changes (Shabanah et al. 2010). (Hatzilygeroudis et al. 2012) implied that games provide an immersive environment where students can

grasp computing concepts and algorithmic steps without conscious effort. The paper describes that the researchers have created a game that can assist with teaching in an Artificial Intelligence (AI) course (Hatzilygeroudis et al. 2012). The game involved the map colouring problem to help students learn concepts related to AI, specifically constraint satisfaction, and algorithms such as the arc consistency algorithm. It was revealed that through game-play, students become familiar with difficult constraint satisfaction concepts such as constraint propagation, backtracking, and multiple solution paths, often without realizing the educational depth within the game. When evaluating the game's effectiveness, ten students were placed into two groups. Group A played the game to assist with their teaching, whilst Group B did not play the game. When these two groups of students were tested on their knowledge related to constraint satisfaction problems, group A, on average, correctly answered 56 per cent of the questions, whilst group B only answered 44 per cent correctly. This study highlighted that student knowledge of constraint satisfaction problems improved by playing a game and supports the scope of this project.

2.4.3 Visualizations

Algorithm visualization is identified as a technique that uses graphics, sounds, and animation to explain how algorithms work (Shabanah et al. 2010). Computing education has utilized visualizations since the 1970s (Su et al. 2021). Active engagement with algorithm visualizations is key for learning algorithms and game-based learning tools can provide this active engagement (Shabanah et al. 2010). However, it has been demonstrated that visualisations alone are often not sufficient for successful learning (Su et al. 2021) . For an optimal level of engagement, students need an interactive environment. Actively engaging with activities related to learning objectives can be directly correlated with improvements in learning when compared to passive viewing (Su et al. 2021). Additionally, If visualizations lack natural connections to real-world objects, it can limit the creation of analogies or metaphors (Su et al. 2021). Creating analogies/metaphors is important in data structures and algorithms because they can often be quite abstract to learn about, therefore using an analogy can make learning much easier as learners can compare the algorithms to real world scenarios.

Through research on the effectiveness of existing game-based learning tools and algorithmic visualizations in teaching complex computing concepts like sorting algorithms and constraint satisfaction problems, it's evident that their effectiveness cannot be overlooked. Key features in the existing game-based learning tools include interactive tutorials, different game modes, point systems and real-world metaphors of abstract concepts. Moreover, the role of algorithm visualizations is crucial in aiding algorithm learning, but it's important to note that visualizations which encourage active engagement are key to maximizing learning outcomes. Incorporating these elements into the game-based learning tool aims to create an immersive and effective learning environment that caters to the learner's needs.

3 Analysis/Requirements

This chapter is concerned with creating the requirements, based off the project's aims, that will shape the design and functionality of the game-based learning tool. The success of the game-based learning tool is directly related to a comprehensive understanding of the functional and non-functional requirements that the project entails. The background research will influence these requirements and dictate the features integrated into the project, thus ensuring engaging game-play. The requirements were created using the MOSCOW technique which is a technique that is used for prioritizing requirements into four different categories (Hudaib et al. 2018).

These categories are referred to as:

- **Must Have** requirements must be in the project, where a failure to deliver these requirements results in the whole project being a failure.
- **Should Have** requirements should be in the project and can be described as the second most important features to prioritize. They are features that are not critical for launch but still have a high value to the users.
- **Could Have** requirements could be in the project and can be seen as the third most important features. These features are usually desirable but not necessary for the success of a project.
- **Won't have** requirements are the least important requirements that provide little value to the project and are usually, but not always, ignored.

3.1 Functional Requirements

Functional requirements can be seen as the backbone of the game-based learning tool and include the functional features and capabilities of the tool. These requirements can be related to how the user interacts with the system and how these interactions align with the learning objectives.

3.1.1 Must Have

- **Game Mechanics**
  Using the constructivist learning approach, the game mechanics must trigger students to use their knowledge of sorting algorithms to complete problem-solving tasks in a gamified environment. This is the most important requirement of the whole game, and students must be taught abstractly about the sorting algorithms within the game. The principles outlined in the background research, such as interactivity, real-world representation, freedom and exploration, must be used to captivate the user's interest in learning the sorting algorithms, creating a more immersive environment.
- **Algorithms Content**
  The game must contain at least 3 sorting challenges for users to learn. Due to the time constraints which the project presents, 3 sorting algorithms seem reasonable as an absolute must.

- **Progressive Difficulty Levels**
  The game must become more difficult as the user progresses, and the sorting algorithms must be taught in order of difficulty. This directly relates to the challenge-response-feedback model in game design, where challenges should match the learner's skill level to maintain engagement without causing frustration.
- **Learning Resources**
  The system must provide learning resources such as pseudocode, hints, or visualizations for algorithmic concepts. As this is a game-based learning tool, students will need the appropriate resources within the game to complete the learning objectives. Providing learning resources corresponds to the importance of interactive tutorials and guidance, as seen in existing game-based learning tools such as Sortko (Boticki et al. 2012) and DeCode (Su et al. 2021).
- **Feedback Mechanisms**
  The sorting challenges must have appropriate feedback mechanisms, a key phase in the challenge-response-feedback model, to guide the user to the solution and help them correct errors. This will allow users to understand if they are approaching the problem correctly and to learn from their mistakes, as seen in Sortko (Boticki et al. 2012).

3.1.2 Should Have

- **Home Page**
  A home page should be included as the main navigation site of the game. It will contain buttons to navigate to different game modes and gamification features within the game.
- **User Registration and Profiles**
  Users should be able to register accounts on the game with unique usernames. The account data will include the users' achievements in the game and performance metrics related to the sorted challenges, which will be contained in a profile section that the users can see. Profiles should provide a personalised experience, as seen in Duolingo (Duolingo 2019).
- **Leaderboards**
  The game should feature leaderboards displaying the top-performing players on each sorting challenge. This adds a social dimension to the game by providing friendly competition that could motivate users to play or reengage with it, as observed in Duolingo's implementation of leaderboards (Duolingo 2019).
- **Badges**
  Badges should be awarded to players based on progress in the game. They should serve as effective incentives to accomplish specific tasks and engage with the game, similar to the gamification feature of achievements, as seen in Duolingo (Duolingo 2019).
- **Various Interactive Modes**
  The game should include different modes to engage users. Providing multiple interactive modes caters to diverse learning preferences and enhances engagement, as seen in DeCode's incorporation of different levels and modes for learning sorting algorithms (Su et al. 2021).
- **Menu**
  A menu should be included when not in the homepage scene to allow users to exit or restart a scene easily. This is helpful because it allows users to restart a sorting challenge if they've approached it wrong or to go back to the home page and play a different game mode.

3.1.3 Could Have

- **User-Generated Visualizations**
  The game could allow users to create their own visualizations. This aligns with the importance of active participation when using visualizations and greatly aids with algorithmic learning.

- **User Feedback System**
  The game could have a feedback system where players can report issues, suggest improvements, or seek assistance.

3.1.4 Won't Have

- **Mobile Support**
  The game is primarily made for PC, so it won't be supported on mobile devices.

3.2 Non-Functional Requirements

Non-functional requirements address the game-based learning tool's performance, usability and overall user experience. It is more concerned with the elements of the game that aren't functional, such as the aesthetic or the system's usability.

3.2.1 Must Have

- **Gamification**
  The gamification features must be designed appropriately and according to the knowledge gained from the background research, as they are essential for sustaining user engagement and motivation.
- **Game Interface**
  The game interface must be intuitive and allow the player to navigate the game easily, without prior use, to enhance usability and promote immersion. The game elements must be clear and user-friendly. Any part of the game which isn't seen as intuitive must have clear instructions.
- **Compatibility/Deployment**
  The deployment must be a .exe file that can be run on Windows.
- **Fantasy**
  The game must follow a certain fantasy that will engage the player and that can be used to create a narrative. Creating a compelling fantasy enhances immersion and engagement, as observed in the background research, where narratives and sensory experiences contribute to a sense of presence in the game.
- **Usabillity**
  The game must be playable by learners with limited knowledge of sorting algorithms. However, having programming knowledge will significantly help the learner understand the game's concept.

3.2.2 Should Have

- **Audio and Background Music**
  The sorting challenges and story mode should contain audio and background music to create a more sensory and immersive experience.
- **Scalability**
  The system should be designed to handle a growing number of users and data.
- **Response/Loading times**
  The response times for user interactions with the game should be low, and the game should load quickly and efficiently

3.2.3 Could Have

- **Customization**
  Users could be allowed to customise their playable characters or profiles to create a more personalised experience.
- **Integration with Social Media**
  Social media could be integrated with the game to allow users to share their achievements or challenge friends to beat them in the leaderboards.

3.2.4 Won't Have

- **Cinematics**
  The game won't have cinematics as they are not essential for game-based learning tools.

4 Design

This chapter explores the design plan for the game-based learning tool, highlighting the steps taken before the actual implementation. The aim was to comprehensively understand the project before bringing it to life. The narrative will unravel the thought process and strategic decisions made to lay a solid foundation for creating a game-based tool.

4.1 Sorting Challenges and Algorithms

After considering all the requirements, it was decided that the game would contain two different types of sorting challenges for each algorithm included in the game: pseudocode and visualization. Having two different types of challenges reduces repetition, which increases engagement. However, the type of pseudocode and visualization challenge depended on the sorting algorithms used in the game. Therefore, the sorting algorithms were to be decided before continuing with developing ideas for the sorting challenges. When deciding what sorting algorithms to include in the game, the first factor to consider was their difficulty, ensuring that the difficulty of the algorithms increased as the game progressed. Additionally, the sorting algorithms were decided based on the concept of which one could be implemented into a visualization challenge. This leads to the conclusion that bubble, quick, and merge sort would best suit the game.

4.1.1 Pseudocode challenges: fill-in-the-blanks and Parson's problems

The approach for the bubble sort challenge involves filling in the blanks within the pseudocode. This design aims to prompt students to draw upon their retention of the bubble sort pseudocode. Given its relatively small amount of pseudocode, this challenge was deemed suitable for bubble sort, where recalling keywords in lines of code is expected to be manageable.

However, when considering quicksort and mergesort, their pseudocode is notably larger. Therefore, fill-in-the-blank methods for these algorithms might be quite cumbersome, and direct recall of missing keywords could prove challenging for users. Therefore, it was decided that Parsons problems would be more fitting for these algorithms. Parson problems are "a type of programming exercise where students rearrange jumbled code blocks of a solution program back into its original form" (Du et al. 2020). These problems are seen as an effective tool for students to practice and improve their knowledge of the code within sorting algorithms. As users rearrange blocks on their screen in Parsons problems, the challenge becomes more about understanding code structure than solely relying on memory. To further assist users, indentation will be used in the blocks within Parsons problems. Feedback mechanisms will be included in all the challenges to indicate if a user has correctly filled in the blanks or reordered the blocks.

4.1.2 Visualization challenges

Each visualization challenge will begin with an unordered list of numbers within and will be represented by real-world objects relevant to the game's fantasy. The design for both bubble sort and quick sort visualization challenges will follow the same approach of illustrating the step-by-step execution of element swaps within the list until an ordered list is achieved. Quick

sort will also highlight once a pivot has been fully partitioned. In the case of the merge sort visualization challenge, users are to split the unordered list up into it's individual elements and merge these back together just as merge sort would.

All the visualization challenges will include feedback mechanisms, highlighting whether a swap (in bubble sort and quick sorts) or a placement (in merge sort) is correct. Thus, the user will be guided to the solution and will have an instructive learning experience.

4.1.3 Choice of game scope based on integration of challenges

The scope of the sorting challenges has been decided. However, these challenges alone are insufficient to make a game-based learning tool successful. Instead, these sorting algorithms should be integrated into a game environment to give the user the full gamified experience. Therefore, the scope of the game itself must be decided. Based on the requirements of the game-based tool, two different ideas were proposed including:

- **Algorithm Racer**
  A multiplayer game where players complete sorting challenges against other players in real-time. The racing fantasy would be approached with each lap corresponding to a player successfully completing a sorting challenge. Therefore, the objective is to complete the sorting challenges as quickly as possible and become victorious by beating the other players in the race. Leaderboards would be added to show the fastest competition times for sorting challenges, adding an extra competitive edge to the game. Badges can be earned for accomplishing specific tasks within the game. The game would also feature distinct modes, including a learn mode for learning the sorting algorithms, a practice mode to practice the sorting challenges, and the main mode, online multiplayer, for real-time interaction with other players. Additionally, the game could introduce team races to encourage collaboration among players.
- **SortQuest**
  Focused more on an immersive single-player gaming experience, it would include a story mode as opposed to a multiplayer mode in the previous idea. In the story mode, players will embark on a solo journey through a medieval fantasy realm, where each land in the story is inspired by one of the sorting algorithms. The narrative would unfold with players exploring the lands, filled with traps and surprises, where sorting algorithms are integrated into this narrative. For example, one challenge could be sorting the keys into the correct order by demonstrating a sorting algorithm and thus unlocking a door allowing the player to progress. The player will be given the role of a controllable character, a knight, to journey through the lands and challenges to restore peace to the world. This engaging narrative could provide freedom and exploration. The game would also include a practice mode that will intensify the sorting challenges by recording performance metrics and presenting them in online leaderboards, thus creating a social dimension within the game. The learn mode would help the players learn the sorting algorithms through pseudocode and visualizations.

When exploring these two options for a game-based learning tool, the SortQuest concept stands out as the preferred choice. The decision to prioritize SortQuest over Algorithm Racer is rooted in providing a more immersive and comprehensive approach that aligns better with the project requirements. While both concepts share common elements, such as the incorporation of learn and practice mode, the main factor in choosing between them was whether a single-player story mode or a multiplayer mode would better cater to the project's specific needs. Therefore, it was decided that a single-player story mode incorporates more of the requirements for this project than a multiplayer mode, and thus, SortQuest will provide an educational gaming experience that transcends conventional boundaries.

4.2 Prototypes

Before the implementation of the project is executed, several prototypes have to be made to create an initial idea of what the game should be like, illustrated in Fig. 4.1. (Deininger et al. 2017) describes prototypes as "visual and tangible tools for communicating ideas, especially during the front-end phases of design, including problem definition and ideation". The software used to create these prototypes is Figma, which allows users to create, charge and test designs for websites and applications. (Figma 2023).

Please note that the learning mode was initially called tutorial mode. However, because the word 'tutorial' in games usually refers to a guide for how to play the game as a whole, it seemed appropriate to rename the mode as learn mode.

4.3 Game Features

In this section, the agreed game features of SortQuest are highlighted based on the background research and the project's requirements.

4.3.1 Fantasy

SortQuest will be set in a medieval fantasy realm inspired by sorting algorithms. This thematic choice will create an immersive user experience and an exciting narrative for Story Mode. It will encapsulate every feature of the game to be designed to suit this medieval theme.

4.3.2 Narrative

The narrative will be most prevalent in Story Mode, where players assume the role of a lone knight navigating through the medieval landscape. The knight will start in bubble sort land after an evil force has taken over the world, and the knight is set on a quest to journey through the other lands of quick sort and merge sort to find the sword that will restore peace to the world. Each land in the story represents a different sorting algorithm; thus, each algorithm's challenges will be integrated. The narrative will directly relate to the sorting challenges, as completing these challenges will solve a problem that the knight encounters in each land. Additionally, the numbers in the sorting challenges will be represented by an object related to the problem the knight is facing, connecting the challenges to the narrative.

4.3.3 Platformer

The sorting challenges in the story mode will be connected through a platformer world with three different lands: bubble sort land, quick sort land, and merge sort land. Additionally, there will be a final scene in the platformer world where the knight enters the castle and obtains the sword to bring peace to the world. A platformer is "a sub-genre of action video games in which the core objective is to move the player character between points in an environment." (Wikipedia 2023). This works perfectly with the scope of the story mode, as the points in the environment can represent where a sorting challenge is to be integrated. In addition, in the platformer world, there will also be various obstacles and traps that can catch the player out. This key feature provides a purely enjoyable element to the game without any intense learning, thus creating a level of replayability.

4.3.4 Interaction

The player and system interaction will ultimately define the gaming experience. The user-system interaction for the game will have simplicity in mind. The game navigation will be done by

clicking clearly labelled buttons. The platformer world will use very simple controls, such as the keypad for moving and space for jumping. The bubble sort pseudocode challenge will use the keyboard for typing, and the rest of the sorting challenges will all be based on using the mouse to drag and drop or click the element specified in the challenge description.

4.3.5 Progressive difficulty

The progressive difficulty within SortQuest will cater to a learning-orientated trajectory. The story mode begins with the relatively straightforward bubble sort land, the easiest algorithm to learn in the game and thus the easiest challenges to complete. Bubble sort land will also include the fewest obstacles and traps and should generally be easy to complete. Quick sort land will contain harder sorting challenges and more traps and obstacles, with moving traps and elevators being introduced. Merge sort land will be the last land the player embarks on and the hardest land to complete. This will be due to more difficult sorting challenges and further complex obstacles presented to the players. Additionally, more advanced moving platforms and a fire trap will be introduced in merge sort land.

4.3.6 Pseudocode and Visualizations

The learn mode will incorporate visualizations and pseudocode explanations. The visualizations will be dynamic within the game and have play/pause capabilities in order that learners play the visualizations at their own pace, creating a more personalized experience. These learning aids can serve as a preparatory resource, allowing players to reinforce their knowledge before entering the story or practice mode. Within the sorting challenges, a hint button will let players look at visualisations when doing pseudocode challenges and vice versa. This strategic design encourages players to actively contemplate the pseudocode from the visualizations or the visualizations from the pseudocode, promoting a deeper level of comprehension.

4.3.7 Feedback

In the sorting algorithms, real-time feedback will be crucial. Utilizing textual cues and colour-coded highlights of red and green will help guide the player towards the correct solutions. For example, if a student splits the initial unordered list in the merge sort visualization challenge wrong, the box containing the wrong element will be highlighted in red. These feedback mechanisms will contribute to a supportive learning environment emphasising understanding over memorization.

4.3.8 Leaderboards

This feature will be the game's main social dimension, motivating players to play the practice mode. Displaying the top 20 times for each sorting challenge will create healthy competition that can motivate players to improve their performance. This enhances replayability, providing a continuous sense of achievement and progress as players can keep competing against other players to gain the best time.

4.3.9 Badges

Badges can serve as incentives for completing specific tasks, adding an extra layer of motivation. For example, a badge could be awarded for completing the story mode bubble sort part. These incentives could encourage players to obtain all the badges as the badges obtained will be shown in the profile section for each player.

4.3.10 User Registration and Profiles

User registration and profiles will create a more personalised experience and power the leaderboards and badge systems. This is because there will be an option to view your profile, including all the best times for each sorting challenge in practice mode and the badges a player has been rewarded. It will also help the leaderboards function better as it automatically uploads the player's score to the leaderboard if they have obtained a high score. As the app doesn't contain sensitive data, the user registration will be authenticated using only a username and password.

4.3.11 Menu

This feature will allow a player to leave a game mode or restart a scene within a game mode. Scenarios in which the menu could be used include, if the player experiences difficulty in one of the sorting challenges and would like to restart it or if they encounter a bug that renders the game unplayable. It should mitigate the chances that a problem in the game will impact them for the rest of the experience.

4.3.12 Style

The overall style of the game will be based on medieval fantasy. This means that every game aspect will be related to the medieval theme, including buttons, audio, narrative, the platformer environment, etc. creating a less boring, intense, and fun learning environment for the players.

4.3.13 Audio

The incorporation of audio elements will heighten sensory engagement. Audio for the knight animations and medieval background music will be added to the platformer world and the sorting challenges. This will create a more immersive sensory experience, as the player will also use the sense of hearing in the game.

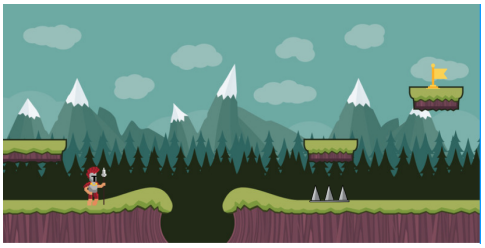
4.4 System Architecture

The game must use a database to store user data and leaderboard data.

As there are only low levels of insensitive data, the architecture for the game will be fairly simple and look like this:

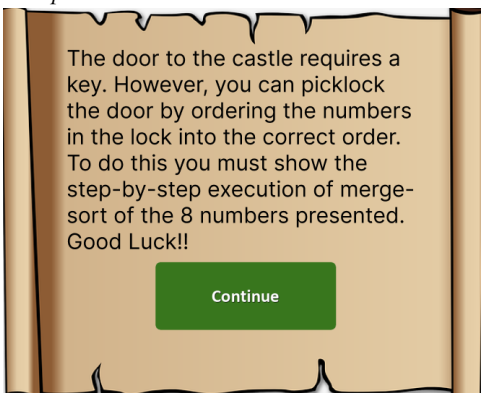
- **Client Side (Front End)**
  Responsible for rendering the game interface and interacting with the player. It will include the game logic, user interfaces, and all other components needed for gameplay.
- **Server Side (Back End)**
  The server side will manage the game's online logic through API calls. However, little code is needed because the server side is very specific to a few features and is absent throughout most of the game. Therefore, it can be integrated into the same code base as the client side for simplicity.
- **Database**
  The database will store user-related data and leaderboard data. The user-related data will contain usernames, badges achieved, and the best times for sorting challenges. The leaderboard data will include 6 different leaderboards, each with the 20 best sorting challenge times with the username as the key and the time as the value.
- **Authentication service**
  An authentication service will be needed to allow users to register profiles and verify their credentials when logging in.

(a) Prototype of the Home Page

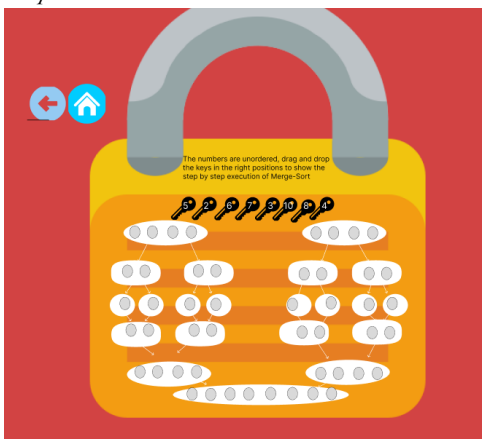


(b) Prototype of Story Mode. The knight represents the controllable character. The spikes represent a trap that will 'kill' the player and cause them to re-spawn at the latest checkpoint. The golden flag represents a checkpoint.

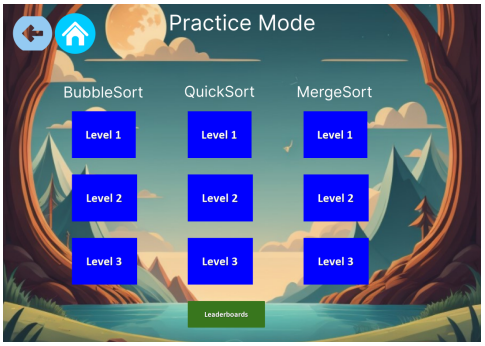
(c) Prototype of a player approaching a challenge in story mode.



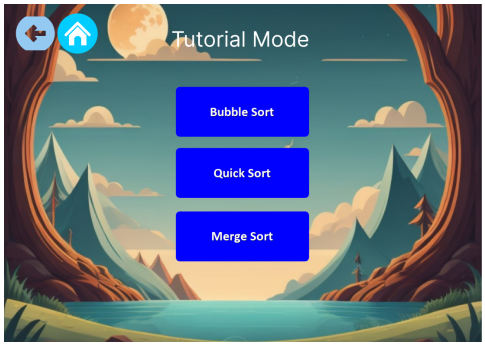
(d) Prototype of a player being informed about the sorting challenge they must complete



(e) Prototype of a sorting challenge. In this case, a Merge Sort Visualization challenge using real-world objects related to the narrative.



(f) Prototype of the practice mode screen before entering a challenge.



(g) Prototype of the initial learn mode screen.

(h) Prototype of the screen before using one of the learning resources for an algorithm.

Figure 4.1: *Prototypes*

5 Implementation

This chapter includes the steps involved in implementing the game-based learning app SortQuest. The first step in the implementation phase is choosing the technology stack for the application.

5.1 Technologies

5.1.1 Game Engine

After evaluating game engines suitable for the project's scope, Unity emerged as the optimal choice. Unity is a versatile game development engine that lets users create immersive 2D and 3D games or experiences that can be deployed across various platforms, including desktops (Windows, macOS, Linux), mobile devices, web browsers, and game consoles. (Unity 2024)

Game objects and scripts are central to Unity's functionality. The scripting is executed in C# and uses object-orientated principles to create the game mechanics. Complementing this, Unity provides a framework called Unity Editor, where scenes can be created, and game objects can be dragged and dropped into them. This means the game is visible as you create it, allowing high transparency between the user and the system.

Unity also contains an asset store which provides a repository of sprites, audio files, backgrounds and more which can be integrated into a creator's game. There is also an Animator within the Unity Editor which allows creators to bring their game objects to life with dynamic animations. Additionally, the physics engine embedded within Unity adds realistic interactions, enhancing the overall immersive experience. Game objects also have an extensive collection of pre-built components that can be added to increase the functionality of game objects.

The option to compile the game into an executable file (.exe) facilitates deployment on Windows. Learners can conveniently download a zip file of the executable and experience the game on their own laptops.

5.1.2 Authentication

Authentication for the game was implemented through the Unity Services API. This is an SDK that can be imported into the project to use different services that Unity offers, in this case, authentication. To activate the authentication, you must log in to the Unity web services and specify the type of authentication needed for the project. The type of authentication picked was username and password, which seemed appropriate as the game doesn't deal with sensitive data, and thus, further verification methods are not needed. The authentication was implemented using a purpose-built script called "AuthManager.cs", which houses all the authentication functionality and is attached to the opening scene containing the sign-in and register buttons. Users were also given the option to continue as guests. However, this meant that the online gamification features would not be available to them.

5.1.3 Backend

The integration of the project backend was achieved through using Firebase (Firebase), a robust application development platform created by Google. Firebase contains a suite of cloud-based backend services and tools which empowers developers to efficiently build, enhance and scale applications. The service which this project will use is the Firebase Realtime Database which is a NoSQL cloud database offering real-time data storage and synchronization capabilities.

Initiating the integration of the real-time database involved the creation of a Firebase project within the Firebase dashboard. Subsequently, the Unity project was added to the Firebase project, generating an SDK which could be downloaded and configured in Unity for immediate use. A simple initialization within the relevant script sufficed to use the database. As specified in the Realtime Database documentation (Firebase), API calls allowed data to be sent to and from the database. For example, a new user record is dynamically generated upon user authentication, containing essential details such as the username, badge achievements, and performance metrics in sorting challenges.

5.2 Sorting Challenges

Each sorting algorithm within the game features a challenge on pseudocode comprehension and a challenge on algorithmic visualization. These challenges unfold in two separate contexts: a practice mode and a story mode. In story mode, the challenges are preceded by a narrative where the player has to solve the challenge to continue with the story. When entering a challenge from practice mode, players are taken directly to the challenge, including a timer that stops once the challenge is completed. Every challenge has a help button which takes the user to a visualization if on a pseudocode challenge and vice versa.

5.2.1 Bubble Sort

Bubble sort is a simple comparison-based sorting algorithm that compares adjacent elements in an array and swaps them if they are in the wrong order. The process of comparing adjacent elements is repeated until the entire array is sorted. At each iteration, the largest unsorted element is compared to its element on its right (if sorting in ascending order) and "bubbles up" to its correct position. However, despite the simplicity of bubble sort, it is not efficient for large datasets because its time complexity is $O(N^2)$, making it more suitable for educational purposes as opposed to practical applications.

Pseudocode Challenge

The bubble sort pseudocode challenge prompts users to complete missing sections of pseudocode by filling in the blanks. The 'BubbleLevelOne.cs' script facilitates this challenge and organizes its core functionality. Fig. 5.1a illustrates the UI for this challenge.

The functionality is fairly simple: each blank text input field, hiding a portion of pseudocode, is equipped with a listener. As a user inputs text into the blank field, the listeners dynamically evaluate whether the text equals the hidden code. If so, the field will turn green to indicate a correct answer.

When a user completes all the blank fields, a victory sound is signalled, and a continue button appears, inviting users to progress with the story or return back to practice mode, depending on where they entered the challenge from.

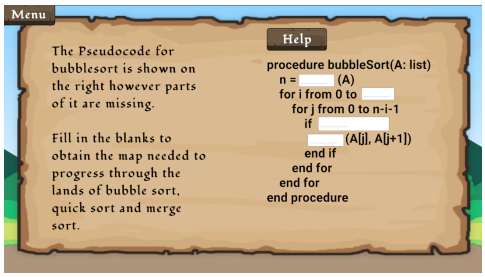
It should also be noted that every user input undergoes whitespace trimming and conversion to lowercase to ensure consistency in evaluating the answers of the text input fields.

Visualization Challenge

The visualization challenge, Fig. 5.1b, for bubble sort aims to illustrate the sorting process by prompting users to show the sequence of swaps involved in rearranging the numbers in ascending order. The challenge design involves seven slots, each featuring a game object representing a piece of wood with a corresponding number. Using wood to represent the numbers aligns with the narrative of rebuilding a bridge to cross over to quick sort land. The functionality of this challenge involves three scripts: "BubbleSortLevelTwo.cs", "WoodSlot.cs" and "WoodDragDrop.cs". In the Unity E ditor, the game objects containing the numbers are dragged to a public game object list called woodNumberArray within the "BubbleSortLevelTwo.cs" script, which allows the tracking of the user swaps.

The "BubbleSortLevelTwo.cs" script contains most of the challenge's functionality. It begins by applying the bubble sort algorithm to the unordered list of numbers visible within the challenge and saving the list's state each time a swap occurs to a list of lists called history. The script then uses a method called "UpdateWoodArray()" linked to the "Check" button, which verifies the correctness of the user's swaps by comparing them against the saved list from the history of each swap generated during the bubble sorting. It does this by iterating through the game objects containing wood, updating the woodNumberArray to reflect the user's swaps. If a number is incorrectly positioned, the corresponding slot is highlighted red; otherwise, it turns green. Completing all swap operations in the correct order triggers the victory sound and reveals the continue button.

"WoodDragDrop.cs" implements the swapping of elements through drag-and-drop interactions using Unity's drag-handler interfaces. This allows users to drag and drop elements into the desired slot to swap the elements. "WoodSlot.cs" complements this functionality by ensuring that the elements snap into position upon being dragged, with the target slot becoming their parent slot.



(a) Bubble Sort pseudocode challenge with fill-in-the-blanks

(b) Bubble Sort visualization challenge including feedback mechanism

Figure 5.1: *Bubble Sort Challenges*

5.2.2 Quick Sort

Quicksort is an efficient divide-and-conquer sorting algorithm that works by selecting a pivot element from the array and partitions the other elements into two sub-arrays according to whether they are less than or greater than the pivot. (In this game, the pivot is partitioned by selecting the last element of the list.) Quicksort then involves recursion to sort the sub-arrays using the same partitioning method. This process continues until the entire array is sorted. The pseudocode for quicksort is therefore split into two parts quicksort() and partition(). It has a time complexity of O(nlogn), making it a fast sorting algorithm that is suitable for use in applications in the real world.

Pseudocode Challenge

The pseudocode challenge for quick sort, Fig. 5.2a, requires users to rearrange jumbled pseudocode blocks into the correct order using up and down toggles connected to the blocks. This problem-solving approach, as mentioned in the design section, is known as parsons problems. The functionality for this challenge is managed by the "ParsonsProblems.cs", "QuickSortLevelOneCheckerPartOne.cs" and "QuickSortLevelOneCheckerPartTwo.cs" scripts.

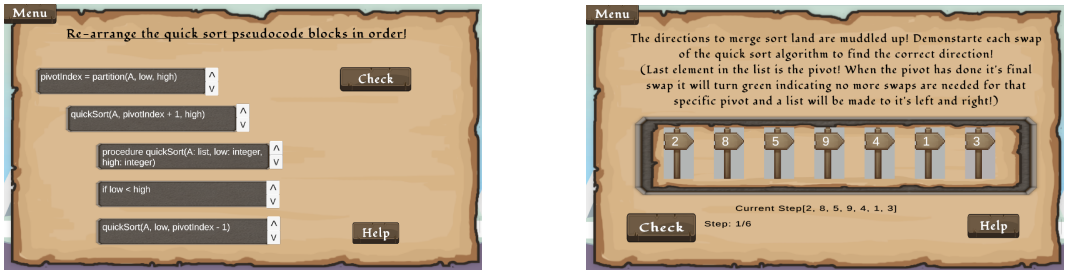
"ParsonsProblems.cs" controls the block-switching functionality, with listeners for up and down buttons attached to each block. Clicking these buttons swaps the text in the block with the text in the target block. The choice for swapping the text instead of the block itself is to maintain consistency with the indentation of the blocks. This ensures that the visual structure of the pseudocode remains intact during rearrangement.

"QuickSortLevelOneCheckerPartOne.cs" and "QuickSortLevelOneCheckerPartTwo.cs" scripts serve the same purpose, checking the order of text blocks. However, while part one assesses the quicksort pseudocode, part two evaluates the partition pseudocode. These scripts are used when the check button is clicked and iterates through the code blocks, comparing the text against an expected order list. If the text is correct, users progress to the next phase; otherwise, they receive text feedback highlighting its incorrect order. If part one evaluates the code blocks as correct, then the challenge continues to part two, and if that is correct, then the user has completed the challenge.

Visualization Challenge

The visualization challenge for quicksort, Fig. 5.2b, is similar to the bubble sort visualization challenge, as a user must demonstrate the quick sorting process by illustrating the sequence of swaps necessary to arrange numbers in ascending order. The challenge involves seven slots, each containing a game object with a number and represented by a direction sign. This representation aligns with the narrative that the directions to merge sort land are muddled up and the user has to complete the challenge to fix the directions. The functionality of this challenge is within three scripts: "QuickSortLevelTwo.cs", "WoodDragDrop.cs" and "WoodSlot.cs". Given that the functionalities of "WoodDragDrop.cs" and "WoodSlot.cs" were elaborated upon in the bubble sort section, they will not be discussed here. In the Unity Editor, the game objects containing the numbers are dragged to a public game object list again, allowing the tracking of user swaps.

"QuickSortLevelTwo.cs" serves as the main class for functionality, it starts with quicksorting the unordered list of visible numbers within the challenge and saving the list's state each time a swap occurs to a list of lists called history. The script includes a method called "UpdateDirectionArray" linked to the check button. The script essentially mirrors the "BubbleSortLevelTwo" script besides using a different sorting algorithm and slot highlighting technique. In this script, a green slot indicates a fully partitioned pivot that should remain stationary, with elements to its left and right forming separate subarrays.



(a) *Quick Sort Pseudocode Challenge* ***(b)*** *Quick Sort Visualization Challenge*

Figure 5.2: *Quick Sort Challenges*

5.2.3 Merge Sort

Merge sort is another efficient divide-and-conquer algorithm that recursively divides an unsorted list of numbers into smaller sublists until it consists of single-element lists, then merges them back together in a sorted manner by comparing the sublists. The pseudocode for merge sort is, therefore, split into two methods: merge() and mergesort(). Merge sort's time complexity is O(nlogn), making it an efficient algorithm that can be used for applications in the real world. Additionally, merge sort is stable, which means it preserves the relative order of equal elements, making it optimal over quick sort.

Pseudocode Challenge

The pseudocode challenge for mergesort closely resembled quicksort, differing only in the specific pseudocode block checker instructions, Fig. 5.3a. Therefore, the same script, "ParsonsProblems.cs" is used to facilitate code swapping. "MergeSortLevelOneCheckerPartOne.cs" and "MergeSortLevelOneCheckerPartTwo.cs", are used to validate the order of pseudocode blocks. Part one verifies the order of the mergesort pseudocode blocks, while part two ensures the correct order of the merge pseudocode. Given the comprehensive discussion of "ParsonsProblems.cs" and the code checker scripts in the quicksort pseudocode challenge section, further elaboration is unnecessary

Visualization Challenge

The visualization challenge for merge sort, Fig. 5.3b challenges users by splitting an unordered list of numbers into individual elements and then merging them back together in sorted order, mirroring the behaviour of the merge sort algorithm. The challenge involves eight slots, each represented by a key game object with a corresponding number. The choice of using keys aligns with the narrative, where sorting the keys is essential to opening the castle and progressing to the final scene. The implementation involved three scripts: "MergeSortLevelTwo.cs", "KeyDragAndDrop.cs" and "KeySlot.cs".

"MergeSortLevelTwo.cs" applies the merge sort algorithm to the visible unordered list of numbers, saving each number in the execution order using callbacks. These callbacks capture the intermediate state of the numbers within each sublist and store them in the "mergeHistory" list. The script also includes a method called "KeyCheckUpdate()" linked to the "Check" button which verifies if the user has correctly split and merged the game objects. It iterates through each slot, comparing the numbers within each slot's key and evaluating if the number within is in the same order as the "mergeHistory" list. Correct slots are coloured green, incorrect ones are coloured red. The continue button will appear if all slots are correct, and the victory sound will play.

"KeyDragAndDrop.cs" includes additional functionality compared to the "WoodDragAndDrop.cs" script. The extra functionality is related to the game objects being duplicated, and then moving the original key as oppose to just swapping because the user is to demonstrate the full visualization process. The "AllSiblingsOcuupied()" method checks if each slot in a sublist is occupied allowing users to change the key's position without duplicating it. Additionally, only non-duplicate keys can duplicate themselves, preventing incorrect answers caused by excessive duplication.

"KeySlot.cs" mirrors "WoodSlot.cs" but incorporates the "KeyDragAndDrop.cs" component as opposed to the "WoodDragAndDrop.cs" component. In hindsight, consolidating these classes would adhere better to the "Don't Repeat Yourself" software principle.

(a) Merge Sort PseudoCode Challenge *(b) Merge Sort Visualization Challenge*

Figure 5.3: *Merge Sort Challenges*

5.3 Story Mode

In the narrative-driven component of the game, three different realms were created, each representing a sorting algorithm, alongside a final scene. These realms were constructed within a 2D platformer environment, allowing users to control a player through interactive and challenging experiences.

5.3.1 2D Platformer Environment

The immersive 2D environment as seen in Fig. 5.4 serves as the backdrop for the story mode, letting users have the freedom to navigate and explore the environment whilst overcoming obstacles and traps and completing sorting challenges along the way.

Tile-Palette

The game environment was constructed within Unity using its 2d tile-palette feature. The tiles' designs were imported from the Unity Asset Store, and once configured into the tile palette, they could be freely painted into the scene. The palette was given the name "terrain" and was used within an empty game object to manage the functionality of the tiles. To ensure collision detection between game objects and the tiles, various components were added to the empty game object, such as tilemap colliders and static rigid bodies, which enabled objects to interact realistically with the environment. Additionally, a distinct "Ground" layer was created and applied to the terrain, enabling functionalities in scripts, notably allowing the user-controlled knight character to execute jumps only if in contact with the ground layer.

Playable Character

The playable character in the game is represented by a 2D knight sprite obtained from the Unity Asset Store, complete with animations for various actions such as falling, idle, jumping, running, and death. These animations were configured using Unity Editor's animation and animator windows. The knight character was given a rigid body and a box collider to enable movement across the tiles. Movement controls were simple, using the left and right arrow keys, or alternatively, the "A" and "D" keys for left and right movement, and the space bar for jumping.

The "PlayerMovement.cs" script encapsulates the knight's core functionality. The "ButtonPressed()" method checks for user input to initiate movement and jumping, adjusting the character's position using vectors. Additionally, pressing the space bar triggers the knight to jump, accompanied by the jump sound effect. The "UpdateAnimationState()" method determines the appropriate animation state based on the knight's coordinates changes. For example, a change in the y-coordinate indicates falling or jumping, while changes in the x-coordinate indicate movement to the left or right. No change in coordinates indicates that the knight is considered idle. Unity's Update() method continuously checks these methods in each frame. The

"IsGrounded()" method verifies if the character is in contact with the ground layer and is used when the player tries to jump. Various parameters, such as move speed and jump force, were serialised to allow real-time adjustments during game-play, allowing the perfect values for them to be picked. Serialisation in this context means allowing the developer to edit values in the Unity Editor.

Traps

Various traps, including spikes, rotating saws, and fire, were strategically placed throughout to make the game more challenging. Each trap was introduced progressively as players advanced through the different lands. These traps were imported from the same package containing this game's tiles.

The main functionality of the traps was coded in the "Death.cs" script, which used box colliders to detect collisions between the knight and the traps. Upon collision, triggered by the player's movement, the knight would die and respawn at the latest checkpoint. A distinct tag, "Trap", was assigned to each trap, which enabled detection, ensuring the game object the knight collided with is indeed a trap. The script also facilitated additional features such as playing the death sound and animation when the knight hits a trap. Furthermore, a mechanism was implemented to ensure that if a player fell off a platform, they would automatically respawn after a 5 second interval.

Introducing dynamic traps such as the rotating saws required specialized scripts to control their behaviour, The "Rotate.cs" script served to make the saw continuously rotate while the "WayPointFollower.cs" script allowed saws to move between two points.

Obstacles

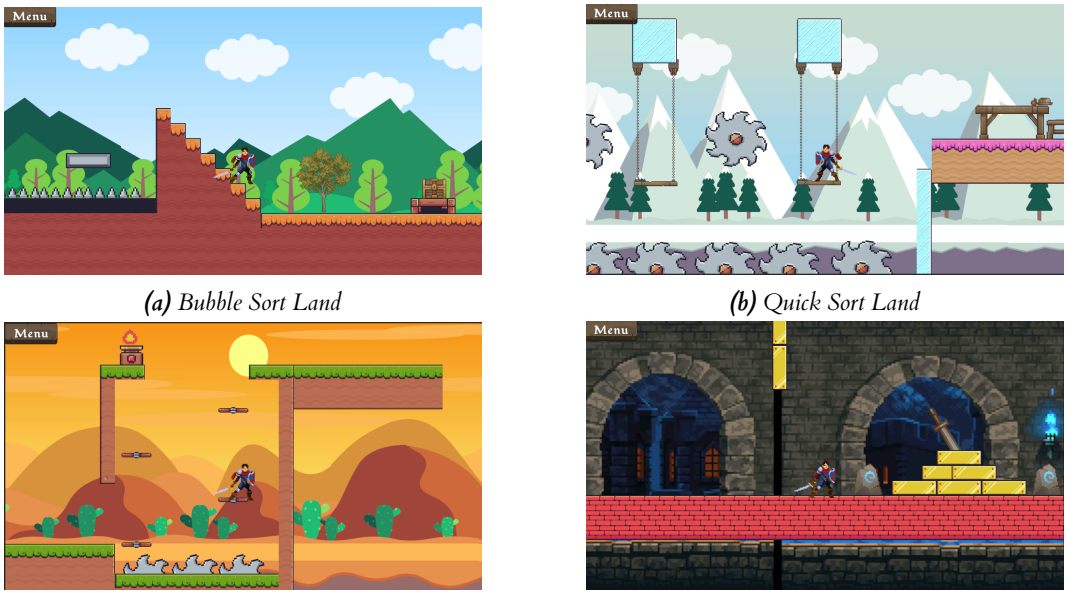
Numerous obstacles were strategically placed in the 2D environments to enhance the game's enjoyment and level of difficulty. These obstacles initially took form through the tile palette, offering intricate pathways requiring precise movement/jumping and advanced into more sophisticated obstacles such as moving platforms and elevators. The "WayPointFollower.cs" script was used again to move platforms between two points, while the "StickyPlatform.cs" script ensured that players were stuck to the platforms during transit. Moreover, the environment was populated with diverse game objects to enrich its visual appeal and create a more immersive experience.

Parallax background

A parallax (moving) background feature was integrated to heighten the game's immersion. This involved importing a package of diverse background environments from the Unity Asset Store, specifically designed with layers for parallax effects. Parallax backgrounds enable a dynamic visual experience by assigning different movement speeds to each layer. To implement this, the script "Parallax.cs" was developed and attached to each background layer. The script uses a float parameter called the "parallaxEffect" to control the background's movement speed relative to the scene's main camera. Consequently, by adjusting the parallax values for each layer, a moving background could be made relative to the player's movement (as the main camera is attached to the knight).

Checkpoints

Checkpoints were placed within the 2D environment and were activated each time a player entered a new land or completed a sorting challenge. These checkpoints served as respawn points, ensuring that if a player encountered a trap or failed an obstacle, they would return to the nearest activated checkpoint. The functionality of checkpoints was coded in two scripts: "Checkpoint.cs" and "InitiateLevel.cs". The "Checkpoint.cs" script was attached to each checkpoint game object, which stored the player's respawn locations and ensured checkpoints could only be activated once. "InitiateLevel.cs" prevented challenge reactivation upon player respawn.



(a) Bubble Sort Land

(b) Quick Sort Land

(c) Merge Sort Land

(d) Final Scene

Figure 5.4: *Platformer World*

5.4 Practice Mode

In practice mode, users engage with the same challenges as in story mode but under timed conditions. A script attached to each challenge button in practice mode lets the system distinguish if the user is coming from practice mode or story mode. It configures the challenge accordingly, activating the timer game object to track the elapsed time. The implementation of this is dealt with by the "Timer.cs" script, which is responsible for initiating the timer upon starting the challenge and ensuring its continuous operation until completion. Additionally, the script includes helper methods for precise time formatting. When a user completes a practice challenge with a high score in terms of time, the user's achievement is recognised through the publication of their completion time in their profile. This process involves the execution of the "publishTime()"method, which transmits the scene name and completion time to the "setScore()" method, which updates the user's profile and, if applicable, the leaderboard with the new high score.

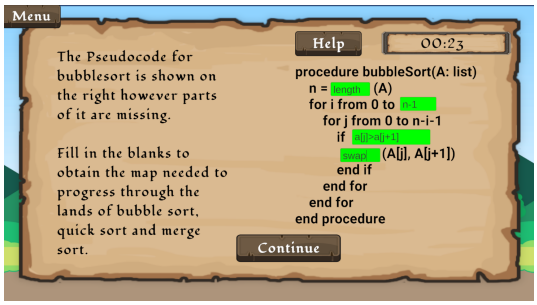


Figure 5.5: *Bubble Sort Pseudocode Challenge Done In Practice Mode (Timer can be seen in top right corner)*

5.4.1 Leaderboards

The functionality of the leaderboards was used across three scripts: "Leaderboard.cs", "Timer.cs" and "LeaderboardEntryUI.cs". As mentioned above, "Timer.cs" uploaded the user's best completion times to the leaderboard following each challenge. "LeaderboardEntryUI.cs" facilitated the correct visualization of leaderboard data, ensuring clarity in presenting position, name, and completion time. "Leaderboard.cs" provided the actual display of leaderboards for each challenge using Firebase calls to reference the leaderboard data and showcase the top 20 positions for the leaderboard selected from the practice mode menu (Fig. 5.6).

Figure 5.6: *Bubble Sort Pseudocode leaderboard*

5.5 Learn Mode

In the learn mode, users are presented with the option to explore the pseudocode or visualization for each of the sorting algorithms (Fig. 5.7a and Fig. 5.7b).

The pseudocode for each sorting algorithm required minimal effort to implement into the game and was simply just copying and pasting the pseudocode for each algorithm into text fields.

In contrast, developing the visualizations was one of the more complex areas of the code base and required scripts for them to work. These scripts were called "BubbleSortVisualization.cs", "QuickSortVisualization.cs", and "MergeSortVisualization.cs" and were able to craft dynamic visualizations of the sorting algorithms. These scripts used Coroutines, which are a Unity feature that allows the execution of code across multiple frames instead of just one. They are particularly helpful with animations/transitions. For example, the visualizations for bubble sort and quick sort dynamically manipulated the positions of the elements on the screen to visualise the sorting process. However, due to the complexity of merge sort requiring game objects to be duplicated and moved, a different approach was adopted, involving the activation of game objects instead of moving them dynamically. Each visualization script included additional methods to support functionalities such as play, pause and reset which are essential to enrich the learning experience.

(a) *Quick sort pseudocode in learn mode* ***(b)*** *Quick sort visualization in learn mode*

Figure 5.7: *Example of a sorting algorithm in learn mode (Quick Sort)*

5.6 Profiles and Badges

The user profiles in the game are accessible via the MainMenu scene by clicking the practice button (Fig. 5.8). Here, users can view their earned badges and the best times achieved in each practice mode challenge. When a user registers, both types of data are initalised to zero. Badge functionality for setting the badge data is managed by the "SceneSelector.cs" script, which rewards users with badges such as the "Bubble Sort Expert" badge, which is achieved by entering the "QuickSortStory" scene from the "BubbleSortStory" scene. This achievement triggers Firebase API calls, updating the user profiles accordingly. The best times are recorded by the "Timer.cs" script as mentioned above. The displaying of the data is managed by the "UserProfileMainMenu.cs" script in the MainMenu scene. This script contains the methods: DisplayPlayerName(), DisplayUserScores(), and FetchAndDisplayBadges(). Additionally, a FormatTime() method ensures the text for the corresponding challenge is set to N/A if the user hasn't attempted it, which is known if the time is 0 seconds. Another method, SetBadgeTransparency(), would set the badge transparency to 255, making it opaque, if the badge value is one, meaning they have achieved the badge. Otherwise, the badge will stay transparent.

Figure 5.8: *Profile including badges and best times. User has only completed one practice mode challenge, and four badges*

6 Evaluation

6.1 Testing

The initial testing approach for SortQuest involved using the Unity Test Framework. However, attempts to create references to assemblies to integrate the scripts for testing purposes resulted in various bug and errors in the application, leading to the abandonment of this approach after extensive debugging efforts.

Subsequently, an alternative testing method using AltTester(Alt) was explored. Despite thoroughly following the documentation, issues persisted with the configuration of AltTester in Unity, and after debugging for a considerable amount of time, this approach had to be abandoned as well.

Regrettably, due to testing being done at the end of the project, the time constraints and the focus on dissertation writing resulted in further exploration of testing methods becoming unfeasible. However, while automated testing was not feasible, extensive personal playtesting and feedback from the questionnaire participants yielded no significant bugs or issues in SortQuest

6.2 User Evaluation

This section involves recruiting ten computing students to play SortQuest and subsequently gathering their feedback through a structured questionnaire (see Appendix A.4 for the full set of questions). The questionnaire covers various topics, such as the participants' algorithmic knowledge, their perceptions of the system's usability, and their interaction with the gamification features embedded within the game. This feedback serves as a key component in evaluating the extent to which SortQuest aligns with the predefined goals and requirements set out at the beginning of the project.

6.2.1 Algorithmic Knowledge and Teaching Effectiveness

The initial section of the questionnaire involved participants rating their confidence levels regarding each sorting algorithm before playing the game. Specifically, participants were prompted with the following question: *"Before playing the game, how confident do you feel in your knowledge of the Bubble Sort algorithm? 1 - Being not confident at all and 5 - Being very confident."* , where Bubble Sort is replaced with Quick Sort and Merge Sort in the questions that came after.

Following answering these questions, participants were given detailed instructions on how to play all aspects of the game. They were then asked the same question as before but replaced the word "Before" with "After" as they had just played the game to assess whether their confidence levels had increased.

Change in confidence levels

As depicted in Fig. 6.1, noticeable enhancements in confidence levels are observed across participants after playing the game. Specifically, 8 participants exhibited increased confidence levels in bubble sort, merge sort, and 7 participants in quick sort. It is also positive that only one response showed a negative change in confidence levels which was quicksort. These findings

reflect one of the project's main motivations: effectively teaching students sorting algorithms using a game-based learning tool.

Table 6.1 presents the average percentage increases in confidence levels for each algorithmn. This computation involved comparing each participant's initial and final confidence level for every algorithmn. For instance, Participant 1 experienced a 100 per cent increase in confidence for bubble sort, as their confidence level increased from 2 to 4. After computing the percentage increases for all participants and algorithms, the values were summed and divided by 10 to obtain the mean percent increase in confidence levels.

As depicted in Table 6.1, the mean per cent increase rises with the difficulty of the algorithm. This trend suggests that participants might initially exhibit lower confidence levels in algorithms like quick sort and merge sort compared to bubble sort. However, after engaging in the game, their confidence levels appear to increase, indicating an improvement in confidence across all algorithms.

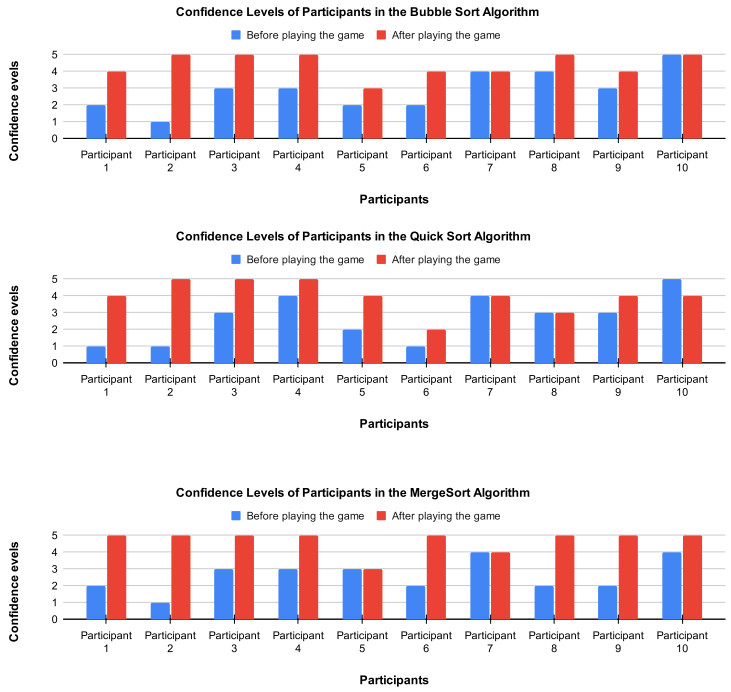


Figure 6.1: *Change in Confidence Levels for Bubble Sort, Quick Sort, and Merge Sort algorithms*

Sorting Algorithm	Mean Percent Increase
Bubble Sort	84.17%
Quick Sort	105.03%
Merge Sort	115.8%

Table 6.1: *Mean Percentage Increase in Confidence Levels*

Qualitive analysis of algorithmic knowledge

After assessing participants' algorithmic knowledge pre and post-game, they were prompted: "If your confidence level in any of the algorithms didn't change, why do you think that is?". Responses were received from half the participants and included various insights. For example,

some participants expressed that they were already familiar with the algorithms, mentioning that the game was a helpful refresher rather than a significant learning experience. One participant stated that they found quicksort quite challenging, thus their confidence level didn't change. Another response felt like there was a lack of base material to teach algorithms, highlighting that the quality of materials in the learn mode could be increased to aid the student using the game-based learning tool.

Teaching effectiveness

To further assess the project's effectiveness in teaching sorting algorithms and its ability to engage users compared to traditional methods, additional questions were made:

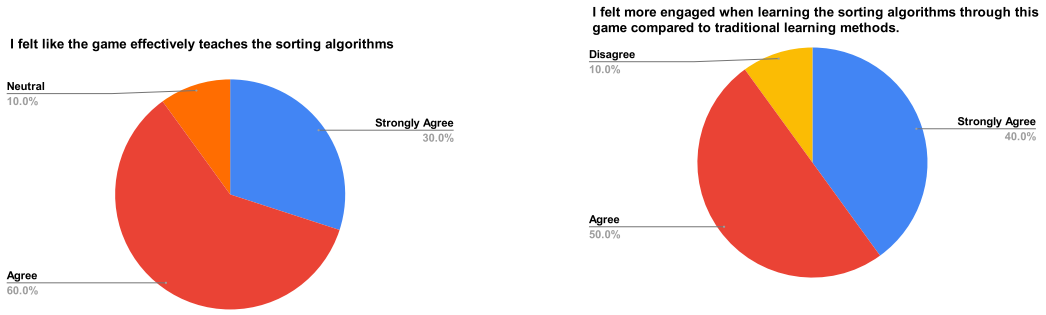


Figure 6.2: *Teaching effectiveness and engagement*

The results, shown in the figures above, demonstrate a strong consensus among participants. All but one respondent agreed with the project's effectiveness and engagement levels, underlining the project's successful achievement of its aims in these areas.

6.2.2 General System Usability

The creator of the System Usability Scale (SUS) (Brooke 1995) describes it as "a simple, ten-item scale giving a global view of subjective assessments of usability." This implies a comprehensive assessment of all usability aspects within the game-based learning tool, which are closely related to the non-functional requirements of the project. Participants are prompted to respond to likert scale questions spanning from "strongly disagree" having a score of 0 to "strongly agree" having a score of 4. Figure 6.3 shows the results for each system usability question from the 10 participants.

Calculating the SUS score

Analysing the participant's responses to the System Usability Scale (SUS), questions provide insight into the system's overall usability. SUS scores, ranging from 0 to 100, are computed by summing the score contributions from each statement, where scores range from 0 (strongly disagree) to 4 (strongly agree). Adjustments are then made by subtracting 1 from odd-numbered questions and the even-numbered questions from 5 before multiplying the total by 2.5 to obtain the SUS score. A SUS score above 68 is considered above average according to industry standards. Notably, odd-numbered questions aim to prompt positive responses, while even-numbered questions aim to prompt negative responses, ensuring a comprehensive usability assessment.

The SUS scores for each participant are listed in Table 6.2.

To get the overall SUS score for SortQuest, the scores in Table 6.2 are averaged by adding them and dividing by 10. This results in an overall SUS score of 76.5. This places the system's usability

Participant ID	1	2	3	4	5	6	7	8	9	10
SUS Score	92.5	87.5	87.5	92.5	77.5	80	50	42.5	75	67.5

Table 6.2: *SUS Scores per participant*

above average in industry standards, indicating a high level of usability that aligns with the project's non-functional requirements. However, it's essential to note that while most participants reported positive experiences across various usability aspects, there were some negative responses. Addressing these areas of concern could potentially elevate the system's usability score and ensure a more consistently positive user experience.

6.2.3 Gamification Features

The final section of quantitative research within the questionnaire focused on evaluating the gamification aspects and their alignment with the project requirements. Once again, a Likert scale was chosen as the optimal response method. The questions and responses can be seen in Fig. 6.4.

The feedback received from participants reflects predominantly positive responses, with the majority expressing agreement with the questions. This suggests the project has effectively met its requirements, highlighting its success as a game-based learning tool. However, it's notable that two participants indicated dissatisfaction with the sorting challenges' feedback mechanisms, which could be related to the absence of colour highlighting in the quick sort visualization challenge, as this was reserved for highlighting the pivots. One participant did not enjoy the game; however, in the qualitative feedback, the same participant said the game was too hard and that they could not complete it. This could be seen as subjective as the other participants found the difficulty suitable, highlighting that the game could include different difficulty levels throughout the game to cater for different individual preferences. Furthermore, a participant's disagreement regarding the usefulness of leaderboards in practice mode could be influenced by the absence of a live player base to create that friendly competition. These insights provide valuable opportunities for refinement and enhancement, ensuring a more inclusive and engaging user experience, however, the main takeaway from this section is that in each question most responses were positive.

6.2.4 Qualitative Feedback

Constructive Feedback

At the end of the questionnaire, this question was asked: *"Could you enter any suggestions or improvements that you may have?"* to prompt constructive feedback. Some of the key responses included:

- *"A sprint button would be helpful in the game."*

  - This suggestion could improve the game mechanics and allow the more experienced "gamers" to complete the story mode quicker.

- *"The hitboxes for the damaging things were too wide."*

  - This highlights that the colliders in the traps could be decreased to provide a less frustrating experience.

- *"I would suggest when showing the animation where how the algorithms work in the learning mode to show the pseudocode next to it so that will be helpful during learning."* and *"For learning, I feel like the code and visualization should not be separate. It could highlight the part of the code it was performing to further educate where in the code it was."*

  - These recommendations could enhance the players learning experience by providing more clarity in how the functionality of the code directly relates to the visualizations.

- *"I think the feedback during the quicksort re-ordering task could be more helpful. It wasn't very intuitive to me"* and *"Make the pseudocode blocks go green and red like the other puzzles when in the right place instead of just saying wrong try again. "*

  - These suggestions likely relate to the response where 2 participants disagreed that there were helpful feedback mechanisms in the quantitative results, thus highlighting the need for more helpful feedback in the quick sort visualization challenge and Parson's problem challenges.

- *"Add more checkpoints, having to do multiple jumping puzzles before a checkpoint was difficult"*

  - This suggestion indicates that checkpoints should be placed after completing a tricky obstacle in the story mode as oppose to just completing a sorting challenge.

All in all, these responses are very valuable and underline constructive feedback that can be used to refine and improve SortQuest.

Additional Observations

The final question asked: *"Any other questions/observations"*. Responses included:

- *"gets progressively harder but stays enjoyable. Not confident on sorting algorithms before but I am now. This would be a great way to teach sorting algorithms and I wish I had this when learning them initially."*

  - This observation is very positive and highlights the success of the project.

- *"Not gonna lie, the spikes hit me a few times causing some frustration but the bgm was awesome ! I did notice the menu didn't scale on my screen on the top left where it says 'Menu' but it's really minor. Having network connectivity to be able to play with friends would be amazing"*

  - This observation indicates a bug to do with the menu button when on a larger display, however, the general consensus is that they enjoyed the game and would love to see more multiplayer features.

- *"Again I think you separated the game from the learning. It had not real connection other than the 'lore' that you had to read. You need to make the environment more to what the game is about otherwise they're just going to brute force it to complete the game."*

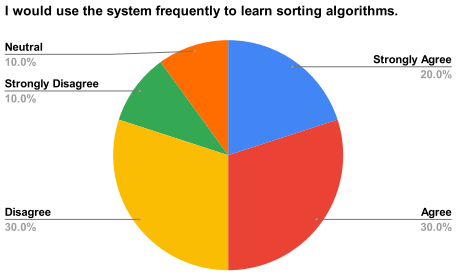
  - This observation implies that the environment which the story is set in should be more connected to sorting algorithms. This is understandable as trying to connect a distinct theme, in this case medieval, to the topic of sorting algorithms proved difficult.

- *"I think as a proof of concept it's a solid game, and the kinds of challenges were pretty interesting. The UI could use some improvements but I imagine there was no time for that, so maybe that could*

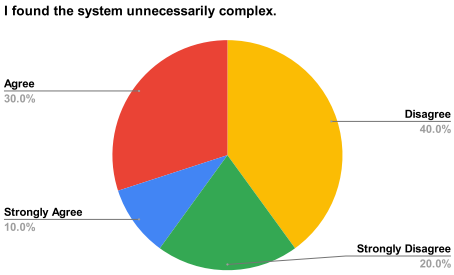
be improved in the future"

- This observation highlights that the participant generally enjoyed the game but would like the interface to be a bit cleaner.

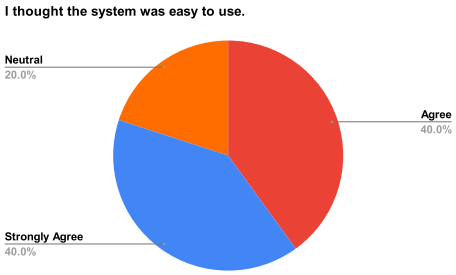
Overall, these observations are generally positive and indicate the project's success.



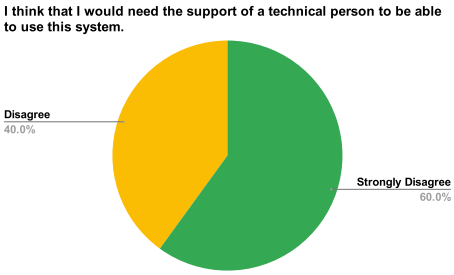
(a) Q1



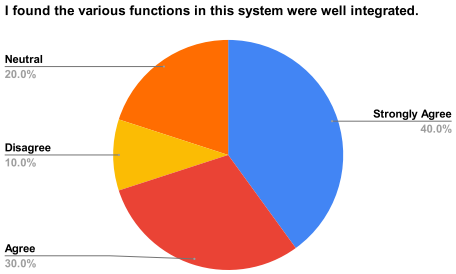
(b) Q2



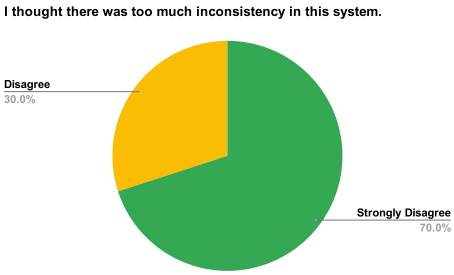
(c) Q3



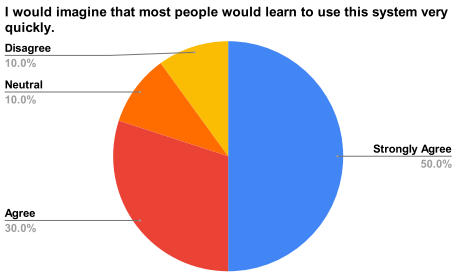
(d) Q4



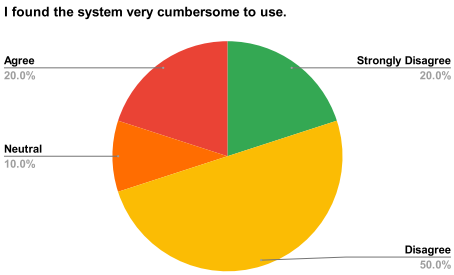
(e) Q5



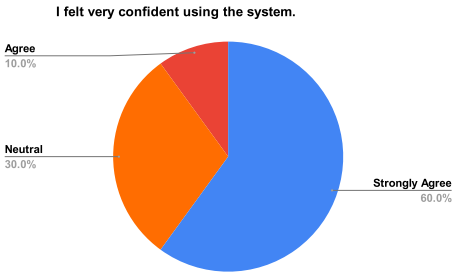
(f) Q6



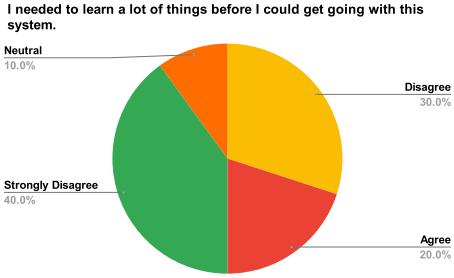
(g) Q7



(h) Q8

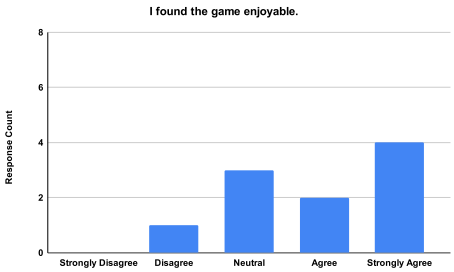


(i) Q9

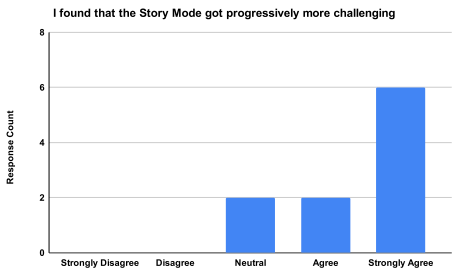


(j) Q10

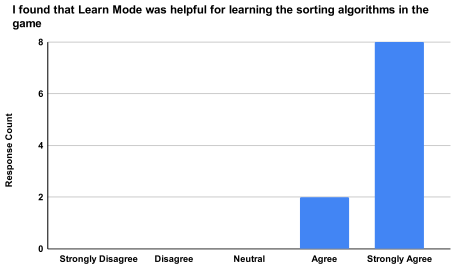
Figure 6.3: *System Usability Scale results*



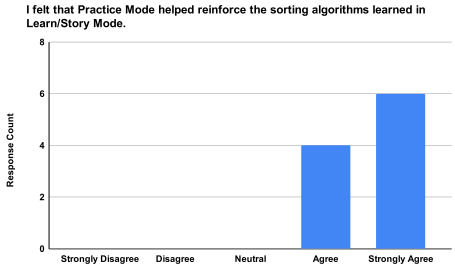
(a) Q1



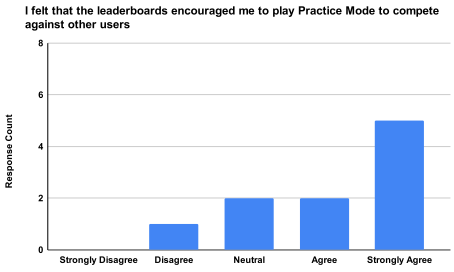
(b) Q2



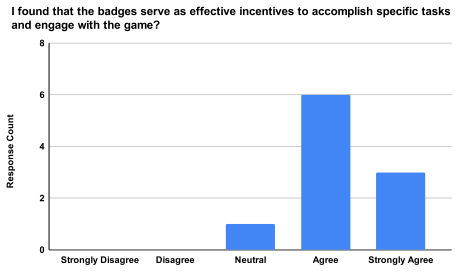
(c) Q3



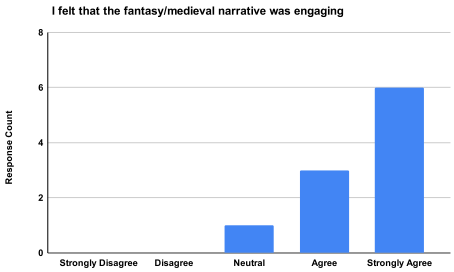
(d) Q4



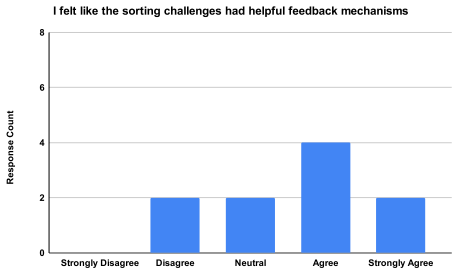
(e) Q5



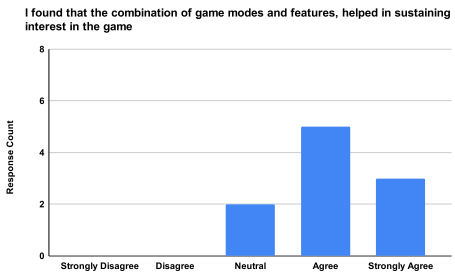
(f) Q6



(g) Q7



(h) Q8



(i) Q9

Figure 6.4: *Gamification questions results*

7 Conclusion

7.1 Summary

This project aimed to teach sorting algorithms in an enjoyable and engaging learning environment whilst incorporating suitable gamification elements into the game. Through thorough background research, it was highlighted what would make this project successful, namely understanding the theory behind game-based learning, which leads to student engagement and learning. Therefore, the scope of the requirements for the project was determined by choosing sorting challenges and gamification features that should be incorporated into SortQuest, focusing on how learning from games can be achieved in computer science classes. The requirements included the gamification features needed to create an enjoyable game-based learning environment: game mechanics, fantasy, narrative, user profiles, badges, leaderboards, audio, a fun platformer world with obstacles and traps, and various interactive game modes. The requirements aided the design of the game, which subsequently led to the implementation of the game using Unity as the main technology. Once the implementation was complete, SortQuest was played on Windows by running it using an executable file.

The project's outcome was analysed through a comprehensive evaluation of teaching effectiveness, the system's usability, gamification elements, and overall user experience, involving 10 participants studying computer science. Teaching effectiveness was initially measured by prompting participants to rate their confidence levels in various sorting algorithms before and after engaging with the game.

Results revealed significant increases, with a mean per cent increase of 84.17% for bubble sort, 105.03% for quick sort, and 115.8% increase for merge sort. Furthermore, 9 out of 10 participants agreed that the game effectively teaches algorithms and enhances engagement compared to traditional teaching methods, underlining the success of the teaching effectiveness and engagement objectives. Qualitative data analysis provided insights into why some participant's confidence levels did not increase in specific sorting algorithms, indicating potential refinements. The system usability scale yielded a score of 76.5, indicating above-average usability, while participant responses on gamification features were predominantly positive, indicating high enjoyment and success of these features. Finally, more qualitative feedback was gathered at the end of the questionnaire, providing constructive insights and suggestions for further refinement. Overall, the evaluation conclusively demonstrated the success of all project aims within SortQuest.

7.2 Future Work

The qualitative feedback gathered from the user questionnaire (Sect. 6.2.1 and Sect. 6.2.4) highlights several areas for potential refinement and enhancement:

- Consider reducing the hitboxes of traps in story mode to alleviate potential frustration among players.
- Explore the integration of visualizations with pseudocode within the learn mode to enhance clarity and comprehension.

- Enhance feedback mechanisms in Parsons problems and quick sort visualization challenges to help guide the users more appropriately to the solution.
- Introduce additional checkpoints in the platformer map to balance focus between completing obstacles and engaging with sorting challenges.
- Strengthen the connection between story mode and learning objectives beyond narrative elements alone.
- Refine the user interface for a more visually appealing and intuitive design.

Furthermore, future work may involve adding more sorting algorithms and addressing the unmet "could have" and "won't have" requirements, such as character customisation, to promote inclusivity and diversity. Additionally, research into accommodating different user needs, such as those with colour blindness or a visual impairment, through features like audio descriptions or colour-blind mode could enhance the game's accessibility and inclusivity.

7.3 Reflection

Upon reflection, this project stands as a remarkable success, meeting all predetermined aims and "Must Have"/ "Should Have" requirements. Throughout the evaluation process, SortQuest achieved generally positive feedback, highlighting it as an effective instructional tool compared to traditional methods. However, potential refinements were identified to enhance further SortQuest and possibly establish a stronger connection between game elements and the theme of sorting algorithms.

Engaging in this project has provided invaluable insights into game development, conveying the gamification component necessary for creating a successful game. This newfound knowledge would help greatly if I were to go down the career path of game development. In addition, the project taught me how to use the Unity framework, a framework I had never used prior, to create immersive and interactive games. Although certain aspects of the project, such as creating the visualizations and their challenges, proved difficult, overcoming these obstacles supported the development of my problem-solving and coding skills. Additionally, coding every aspect of SortQuest from scratch not only sharpened my computing skills but also provided a solid foundation and understanding for coding a computer game. This project allowed me to utilise learning and experience from the past 4 years at university and apply it to create a game that is enjoyable and useful for student learning and acknowledges game-based learning tools.

Overall, this project has equipped me with a robust framework for creating video games, particularly those with educational applications, thus enriching my academic journey and professional aspirations.

A Appendices

A.1 Guide to run software

Copy and paste this link into your browser:

```
https://gla-my.sharepoint.com/personal/2557097h_student_gla_ac_uk/_layouts/
15/onedrive.aspx?id=%2Fpersonal%2F2557097h%5Fstudent%5Fgla%5Fac%5Fuk%
2FDocuments%2FSortQuestFullGame%2Ezip&parent=%2Fpersonal%2F2557097h%
5Fstudent%5Fgla%5Fac%5Fuk%2FDocuments&ga=1
```

Please make sure you are authenticated with a UofG account and:

1. Download SortQuestFullGame.zip above
2. Unzip the folder
3. Start the SortQuest.exe file to play the game.

A.2 Directory Structure

```
SortQuest
    \Assets
        \AltTester
        \Animations
        \Audio
            \BackgroundMusic
            \GameSoundClips
        \Background
            \1_Mountain
            \2_Desert
            \3_Graveyard
            \4_Snow
            \FullBG
        \Editor Default Resources
            \Firebase
        \ExternalDependencyManager
        \Firebase
        \Fonts
            \Fondamento
        \Scenes
        \Scripts
        \Sprites
            \Badges
            \Hero Knight
            \MyPrefabs
            \Pixel Adventure 1
```

```
            \Traps
            \Village Props
        \StreamingAssets
        \Terrain
        \TextMesh Pro
        \UserInterface
            \Fantasy
    \Library
        \APIUpdater
        \Artifacts
        \Bee
        \BuildPlayerData
        \BurstCache
        \com.unity.services.core
        \GridBrush
        \PackageCache
        \PackageManager
        \PlayerDataCache
        \ScriptAssemblies
        \Search
        \ShaderCache
        \SplashScreenCache
        \StateCache
        \TempArtifacts
        \UIElements
    \Logs
    \obj
        \Debug
    \Packages
    \ProjectSettings
    \Temp
        \Burst
    \UserSettings
        \Layouts
```

A.3 Ethics Checklist

School of Computing Science
University of Glasgow

Ethics checklist form for 3rd/4th/5th year, and taught MSc projects

This form is only applicable for projects that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your project.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

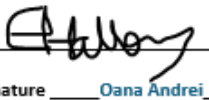
11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. All the data collected from the participants is stored in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Course and Assessment Name Game-based learning for a particular type of algorithms

Student's Name Euan Halliday

Student Number 2557097h

Student's Signature 

Supervisor's Signature Oana Andrei

Date 01/03/2024

A.4 User Evaluation Questionnaire

SortQuest User Evaluation

This survey is in relation to 'SortQuest' which is game-based learning tool meant for teaching various sorting algorithms to students or programmers. As 'SortQuest' is a game-based learning tool, the user must interact with the game, therefore this experiment

evaluates the effectiveness of 'SortQuest' in teaching, the system usability, the gamification features it includes, and the overall user experience.

Before playing the game, answer the questions in this section and you will be giving a list of tasks to complete whilst on the game. After playing the game answer the questions in section three and all subsequent sections.

All data gained will be anonymous and you can withdraw from the survey at any point.

To start the survey you must:

- Consent to taking the survey below
- Not be younger than 16
- Not have any impairments that will limit understanding or communication

If you have any questions, please email me at 2557097h@student.gla.ac.uk or my supervisor Dr. Oana Andrei at Oana.Andrei@glasgow.ac.uk

Here is the link to the game: SortQuestFullGame.zip

Please make sure to login with a UofG account and:

1. Download SortQuestFullGame.zip above

2. Unzip the folder

3. Start the SortQuest.exe file to play the game.

When running SortQuest.exe, your computer might not initially allow it to run because it's an external file, please allow it to run anyway.

If you are not a UofG student, please email me for the game.

If you find any bugs while playing the game which result in you not being able to complete the evaluation: Please either restart the scene if you are in game or email me immediately.

hallidayeuan@gmail.com Switch account

Not shared

* Indicates required question

Before playing the game, how confident do you feel in your knowledge of the Bubble Sort algorithm? 1 Being not confident at all and 5 being very confident. *

1 2 3 4 5

Game Instructions

1. **Account Creation:**
   - Create an account (Account creation is optional; however, it is required for features such as badges and entering leaderboards. Ensure you have an internet connection throughout the game.)
   - Username and password requirements will be shown once on the create an account page.
   - Once account created (or continued as guest) you will be landed at the home menu.
2. **Learn Mode:**
   - Once in the home menu, go to Learn Mode.
   - Explore the visualizations; they can be stopped, resumed, and restarted. Explore the pseudocode, it is for reading purposes only.
   - Go back to the home menu.
3. **Story Mode:**
   - Click Story Mode, check how to play and then play story mode, aiming to get through all the obstacles and traps in the platformer world.
   - Complete the sorting challenges along the way to continue the storyline.
   - Once Story Mode is complete it will take you back to the home menu.
4. **Practice Mode:**
   - After Story Mode is complete, go to Practice Mode.
   - Play at least 1 Practice Mode challenge to record a time to your profile and the leaderboards.
5. **Leaderboard:**
   - Access the leaderboards from the Practice Mode menu.
   - The top 20 scores from each challenge should be visible (if there have been at least 20 entries).
6. **Profile Check:**
   - Go back to the home menu and click on the profile.
   - Check if the best time for any practice challenges completed has been recorded.
   - Ensure that badges have been obtained for completing Story Mode (completing Bubble Sort Land gives you the Bubble Sort Expert badge, completing Quick Sort Land gives you the Quick Sort Expert badge, completing Merge Sort Land gives you the Merge Sort Expert badge).
7. **Complete rest of survey**

Section 2 of 5

Effectiveness of game based learning

In this section, questions are related to finding out the effectiveness of game based learning

After playing the game, how confident do you feel in your knowledge of the Bubble Sort algorithm? 1 Being not confident at all and 5 being very confident. *

	1	2	3	4	5	
Not Confident	○	○	○	○	○	Very Confident

After playing the game, how confident do you feel in your knowledge of the Quick Sort algorithm? 1 Being not confident at all and 5 being very confident. *

	1	2	3	4	5	
Not Confident	○	○	○	○	○	Very Confident

After playing the game, how confident do you feel in your knowledge of the Merge Sort algorithm? 1 Being not confident at all and 5 being very confident. *

	1	2	3	4	5	
Not Confident	○	○	○	○	○	Very Confident

If your confidence level in any of the algorithms didn't change, why do you think that is?

Long answer text

I felt more engaged when learning the sorting algorithms through this game compared to traditional learning methods. *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

I felt like the game effectively teaches the sorting algorithms *

- ○ Strongly Disagree
- ○ Disagree

Section 3 of 5

General System Usability

In this section, questions are related to the usability of 'SortQuest'

I would use the system frequently to learn sorting algorithms. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found the system unnecessarily complex. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I thought the system was easy to use. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I think that I would need the support of a technical person to be able to use this system. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I thought there was too much inconsistency in this system. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I would imagine that most people would learn to use this system very quickly. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found the system very cumbersome to use. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I felt very confident using the system. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I needed to learn a lot of things before I could get going with this system. *

- Strongly Disagree
- Disagree

Section 4 of 5

Gamification Features

In this section, questions are related to the features within 'SortQuest'.

I found the game enjoyable. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found that the Story Mode got progressively more challenging *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found that Learn Mode was helpful for learning the sorting algorithms in the game *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I felt that Practice Mode helped reinforce the sorting algorithms learned in Learn/Story Mode. *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I felt that the leaderboards encouraged me to play Practice Mode to compete against other users *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found that the badges serve as effective incentives to accomplish specific tasks and engage with the game? *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I felt that the fantasy/medieval narrative was engaging *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I felt like the sorting challenges had helpful feedback mechanisms *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

I found that the combination of game modes and features, helped in sustaining interest in the game *

Could you enter any suggestions or improvements that you may have?

Long answer text

After section 4 Continue to next section

Section 5 of 5

Thanks for completing the survey

The main aim of this experiment was to evaluates the effectiveness of 'SortQuest' in teaching, the system usability, the gamification features it includes, and the overall user experience.

If you have any questions or observations please fill in the form below.

If these are urgent, please email me at 2557097h@student.gla.ac.uk or my supervisor Dr. Oana Andrei at Oana.Andrei@glasgow.ac.uk

Any other questions/observations

Long answer text

7 Bibliography

URL https://alttester.com/docs/sdk/latest/pages/commands.html. Accessed 16th March 2024.

M. Barr. *Graduate skills and game-based learning: Using video games for employability in higher education*. Springer, 2019.

L. Begosso, L. Begosso, D. Cunha, J. Pinto, L. Lemos, and M. Nunes. The use of gamification for teaching algorithms. pages 225–231, 09 2018. doi: 10.15439/2018F165.

I. Boticki, A. Barisic, S. Martín, and N. Drljević. Sortko: Learning sorting algorithms with mobile devices. 03 2012. doi: 10.1109/WMUTE.2012.15.

J. Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.

M. Deininger, S. R. Daly, K. H. Sienko, and J. C. Lee. Novice designers' use of prototypes in engineering design. *Design Studies*, 51:25–65, 2017. ISSN 0142-694X. doi: https://doi.org/10.1016/j.destud.2017.04.002. URL https://www.sciencedirect.com/science/article/pii/S0142694X17300273.

Y. Du, A. Luxton-Reilly, and P. Denny. A review of research on parsons problems. pages 195–202, 02 2020. doi: 10.1145/3373165.3373187.

Duolingo. Learn a Language for Free, 2019. URL https://www.duolingo.com/. Accessed 11th March 2024.

Figma. Figma: the collaborative interface design tool, 2023. URL https://www.figma.com/. Accessed 13th March 2024.

Firebase. Firebase documentation. URL https://firebase.google.com/docs/. Accessed 21st March 2024.

I. Hatzilygeroudis, F. Grivokostopoulou, and I. Perikos. Using game-based learning in teaching cs algorithms. 08 2012. doi: 10.1109/TALE.2012.6360338.

S. Hidi and K. A. Renninger. The four-phase model of interest development. *Educational Psychologist*, 41(2):111–127, 2006. doi: 10.1207/s15326985ep4102_4.

A. Hudaib, R. Masadeh, M. Haj Qasem, and A. Alzaqebah. Requirements prioritization techniques comparison. *Modern Applied Science*, 12, 01 2018. doi: 10.5539/mas.v12n2p62.

B. D. H. Jan L. Plass and C. K. Kinzer. Foundations of game-based learning. *Educational Psychologist*, 50(4):258–283, 2015. doi: 10.1080/00461520.2015.1122533.

W. Lim, Y. Cai, D. Yao, and Q. Cao. Visualize and learn sorting algorithms in data structure subject in a game-based learning. pages 384–388, 10 2022. doi: 10.1109/ISMAR-Adjunct57072.2022.00083.

I. K. B. M. Mitchell Shortt, Shantanu Tilak and B. Akinkuolie. Gamification in mobile-assisted language learning: a systematic review of duolingo literature from public release of 2012 to early 2020. *Computer Assisted Language Learning*, 36(3):517–554, 2023. doi: 10.1080/09588221.2021.1933540.

S. Shabanah, J. Chen, H. Wechsler, D. Carr, and E. Wegman. Designing computer games to teach algorithms. pages 1119–1126, 01 2010. doi: 10.1109/ITNG.2010.78.

Y.-R. Shi and J.-L. Shih. Game factors and game-based learning design model. *International Journal of Computer Games Technology*, 2015:1–11, 08 2015. doi: 10.1155/2015/549684.

S. Su, E. Zhang, P. Denny, and N. Giacaman. A game-based approach for teaching algorithms and data structures using visualizations. pages 1128–1134, 03 2021. doi: 10.1145/3408877.3432520.

Unity, 2024. URL `https://unity.com/`. Accessed 20th March 2024.

E. von Glasersfeld. *RADICAL CONSTRUCTIVISM*. Routledge, 1st edition, 1995. doi: 10.4324/9780203454220. URL `https://doi.org/10.4324/9780203454220`.

Wikipedia. Platformer, 2023. URL `https://en.wikipedia.org/wiki/Platformer`. Accessed 13th March 2024.