Politechnika Wrocławska

Faculty of Information and Communication Technology

Field of study: Computer Science

Bachelor thesis

Improving User Experience with the intelligent distribution of trigger points in a procedurally generated dungeon

Dawid Kaluża

keywords:
Educational Games, Maze Generation, Unity,
Active Learning, User Experience

short summary:
Solutions with educational video game using effective learning environment requirements and benefits of active learning thanks to proper user experience design.

Supervisor	dr inż. Jarosław Drapała
	Title/ degree/ name and surname

*For the purposes of archival thesis qualified to: **

a) Category A (perpetual files)

b) Category BE 50 (subject to expertise after 50 years)

** Delete as appropriate*

stamp of the faculty

Wrocław, 2021

CONTENTS

ABSTRACT

The scope of this work is to develop an educational computer game that implements a procedurally generated 3D maze intelligently populated with trigger points that trigger questions when the player collides with them. While traversing the maze, the user will find several trigger points with questions whose correct answers will lead to a better score. The work aims to select appropriate methods for generating mazes and trigger points to improve the user experience, motivate players using standard video game mechanics and use the benefits of active learning.

1. INTRODUCTION

With an **active learning** approach, students can benefit from better analysis and application of knowledge activities. Educational games available on the market usually lack essential functionalities that qualify them as an effective learning environment. We wanted to present a new approach to educational games providing practical learning with the use of a tool for teachers and parents and, at the same time, provide students with an engaging and entertaining video game.

PROJECT AIM

This work aims to develop an educational computer game, "Galaxy Maze," that is attractive for students and promotes active learning. The game should have selected appropriate methods for generating mazes and question triggers to improve the user experience and motivate players using common video game mechanics.

PROJECT SCOPE

The scope of this work is to create an educational game with a procedurally generated 3D maze and objects that will serve as question triggers, provide a fully functional user interface, and implement selected video game mechanics. The student will control the player character, who will find his way inside the maze while answering questions. In addition, the game should be connected to another part of the solution with networking requests and cooperation. Work will also contain a number of user testing methods to validate the effectiveness of proposed solutions.

DEFINITION DICTIONARY

Solution - complete application with all distributed modules, team effort.

Game - part of the solution, a frontend educational game available for students.

Server - the backend application handling that communication between the rest of the modules.

Teacher - an authorized person responsible for creating a scenario, scheduling a game, and controlling the learning result and progress.

Student - targeted player of the game, subject that will be learning.

Class - a group of students for which the scenario we can assign.

Scenario - a set of questions that we can assign to a class, and students can play it when scheduled.

Session - also game playing or single playthrough

Question trigger - also trigger, objective, an object spawned inside the maze and when collided with player character opens a question. Currently, three types of triggers are present: “Key,” “Treasure,” and “Enemy.” Only “Key” is mandatory to collect, while others are optional.

Board - also game board, dungeon, or maze, the 3D board which player will move within and question triggers will be spawned.

Common game mechanics - the mechanics that can be observed in games, usually split into game genre categories, here referring to role-playing in particular.

2. DESCRIPTION OF THE SOLUTION DESIGN

Today, active learning [19] is commonly used to engage students in learning by gamification [9]. However, existing solutions struggle to fulfill all the practical learning environment requirements [12]. To propose a new solution to this problem, we wanted to create an educational game, "Galaxy Maze," that allows for more student engagement than usual educational games. The solution should help teachers introduce a more individual approach for each student's needs without extra work for the teacher and helps control results or learning with easy-to-digest statistics. The solution could be used at school as a part of a lesson or at home as a homework platform accessible easily.

The main goal of this application will be creating it in such a way that allows students to involve more in the game and at the same time allow teachers for easy creation of new quizzes, tests, scenarios. Furthermore, we want to also provide default scenarios with already uploaded questions that will be analyzed by artificial intelligence to adjust the difficulty to student level and help him gain more knowledge from the core curriculum. Finally, the results from scenarios will be processed and then available for the teacher in statistics and summaries.

2.1. SOLUTION REQUIREMENTS

The following functional requirements were proposed to fulfill the solution needs.

— The teacher can log into the management panel.
— The teacher can create, read, update or delete data about classes, students, questions, and scenarios.
— The teacher can schedule a game session by assigning a scenario to a class with a selected time.
— The system will send an invitation email with all required information to play the game to students assigned to a scenario.
— The student can open the game and start a new session after authenticating with a passcode from the email.
— The student can play the game by answering scenario questions.
— The game communicates with the server when the student starts or ends a session or, each time opens, answers a question.
— The system adapts questions difficulty after each received answer from the game.
— The server processes the session data and generates a report and statistics for a teacher.

— The teacher can analyze students' results with easy-to-digest statistics and reports.

Those are general requirements that describe the educational solution in general. The layout of modules needed for those requirements can be also presented in figure 2.1.

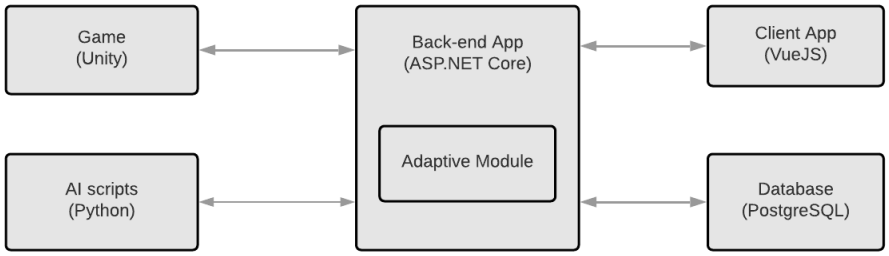

Fig. 2.1. Solution scheme

The non-requirements for the game do not provide any details of how the implementation should look. Those requirements describe the active learning environment and user experience principles that should be met. The game should:

— be attractive to students to enforce the active learning
— be challenging and provide the competitive setting for learning
— have declared steps that the player needs to complete to finish the game successfully
— have a slight learning curve, easy to understand mechanics, and first-time user experience features
— be accessible and have low hardware requirements

It is important to understand that user experience in this case is an wider term than only the ease of use or utility principles but as a game experience and enjoyment. Games User Experience [8] helps create immersion, involvement, and fun. More important in this thesis are the non-functional requirements related to the game design. For now on this will be the main concern of this work.

2.2. ACTIVE LEARNING APPROACH

Active learning is an approach that engages students in an active or experimental process instead of traditional passive learning [19]. Active learning allows students to transfer and apply knowledge better. The effectiveness of active learning is highly dependent on the user experience. The experience design needs to provide a competitive context with a goal and a way to win to engage the students. A game that wants to create an active learning environment needs to:

— have a **purpose**, teach only a limited set of actions or topics

— have **randomness, stress, and rewards**
— be **winnable**, objectives to fulfill
— have an **environment** with a story that the player can be inserted into
— have an **evaluation method** with which player can know if his play-through was good

There are numerous benefits from the active learning approach, but many conditions need to be fulfilled to observe them. The benefits are heavily coupled with the design of the experience. In previous section, 2.1 list of requirements was presented. To evaluate those, we need an evaluation methods like user interviews, play testing, or questionnaires [3].

2.3. STATE OF THE ART SOLUTIONS

Analyzing popular solutions to ensure our solution would provide a new view into educational games and not repeat existing solutions' mistakes. Six popular video games were selected to compare them in two categories of how the solution helps teachers or parents and the students benefit from using it.

1. **Prodigy** [13] - a role-playing 2D game with turn a base fight concentrated on math.
   As a solution, it does provide statistics and a view into student progress, but with no ability to add own content and only math as a learning subject. It also comes with a paid premium version for complete statistics access.
   As for a game, it is a role-playing game similar to the Pokémon video game series with 2D graphics, turn-based fight system where the fight is resolved by answering math questions. Therefore, it is more complex and has a higher entry level. In addition, it has manually generated levels, and role-playing content requires much more work in terms of the development process.

Fig. 2.2. Prodigy gameplay screenshot

2. **Minecraft Education Edition** [10] - an extension to a sandbox game Minecraft with game-based learning addition.

It provides hundreds of standards lessons and some topic-oriented lessons. However, it lacks progress tracking and learning process control. It is also tough to add own content and has high entry-level.

The game is an extension of a viral video game developed over ten years, so it is hard to compete with this size of an application. Still, there is a problem with the hardware requirements as Minecraft is resource-consuming, and not every school or parent can allow for such expenses.

Fig. 2.3. Minecraft Education Edition gameplay screenshot

3. **Big Brain Academy** [11] - puzzle video game published and developed for the Nintendo DS.

   This type of solution does not have any progress tracking or statistics. Furthermore, it is only student-oriented, with no possibility to add new content.

   This game is more about training users' logic and thinking skills, there is no educational part, and the game itself bases on multiple minigames that the player solves one by one. It is also platform-oriented and paid.

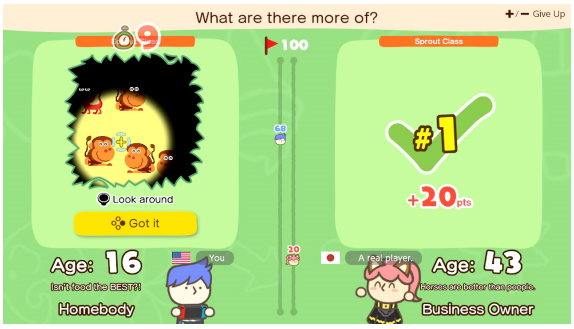

Fig. 2.4. Big Brain Academy gameplay screenshot

4. **National Geographic Challenge** [11] - multiple games oriented around geography and environmental problems. Again no progress tracking, platform-oriented and paid.
5. **Quizlet gravity** [14] is a quiz-like typing game where we need to input the correct answer for the falling block.

Fig. 2.5. National Geographic Challenge gameplay screenshot

It has no progress tracking, but it is generic and has multiple different topics. It is more like a quiz and lacks gaming, so it is less engaging.

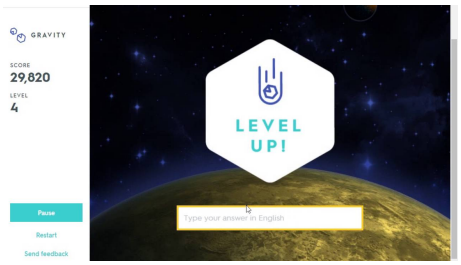

Fig. 2.6. Quizlet gravity gameplay screenshot

6. **Homer** [5] - allows for learning English or math but does not allow for learning progress. It has a monthly subscription, and the teacher cannot add content. The game is fundamental and oriented mainly around the learning part with less gameplay.

Fig. 2.7. Homer gameplay screenshot

The analysis can also be presented in a table. The properties were divided into two tables:

One with the game content-related things like "is the student progress tracking possible" or "can I add my own content to the game" 2.1.

The second one is with more environment-connected topics like "do I need a specific operating system to run it" or "is it difficult for new players to understand the game mechanics" 2.2.

Existing solution	Student progress tracking	Includes popular mechanics	Generic type of knowledge	Adding own content
Prodigy	Yes	Yes	No	No
Minecraft Education Edition	Yes	Yes	Yes	Yes, but it is hard
Big Brain Academy	No	No	No	No
National Geographic Challenge	No	No	No	No
Quizlet gravity	No	No	Yes	Yes
Homer	No	No	No	No
Galaxy Maze	Yes	Yes	Yes	Yes

Table 2.1. Existing solution comparison - Game Content

Existing solution	Hardware require-ments	Platform requirements	Difficulty for new users	Pricing
Prodigy	Low	Browser only	Yes	Free, with premium
Minecraft Education Edition	High	Desktop, con-soles, mobile	Yes	5$ per seat per year
Big Brain Academy	Low	Nintendo only	No	20$
National Geographic Challenge	Low	Consoles only	No	20$
Quizlet gravity	Low	Browsers only	No	Free
Homer	Low	Mobile	No	60$ per year
Galaxy Maze	Low	Browser, mo-bile	No	Free

Table 2.2. Existing solution comparison - Environment

There are, of course, many more games. Still, most of them follow the pattern that they either do not track progress, cannot add content, lack the gaming experience known from popular video games, or have high technological or understanding entry-level.

3. DESIGN OF GAME USER EXPERIENCE

To assure that game is fulfilling all active learning requirements proper analysis of available options is required. This chapter contains the main results of the thesis topic and provides insides to decomposition of user experience requirements into educational, active learning video game.

3.1. DECOMPOSITION OF ACTIVE LEARNING REQUIREMENTS

The created game needs to fulfill the active learning game requirements and at the same time be **generic** and independent from the question topic and **replayable**. While selecting the game environment, it is essential to remember our target player. He is around 10 to 15 years old, and while based on the question source and difficulty can also widen or narrow the age group, we want to make the game mechanics and graphics attractive to students in this age group. Let us look at the requirements proposed before in 2.2 and introduce some solutions one-by-one.

3.1.1. Purpose

The purpose of a game can vary depending on the teacher's scenario. The teacher can decide what type of topic the scenario will cover, and there is little to nothing we can do about it in the game itself. However, all scenarios' questions should have one motive and follow it.

3.1.2. Randomness, stress, and rewards

The environment should change for each attempt to ensure the game's randomness. For example, generating levels instead of manually creating allows for the randomness of the game board and randomness in the objective placement inside of it. We need to encourage the player to explore. Here the stress and reward come into play. The player will search for question triggers that move him closer to the game "win" and introduce the reward for the correct answer. On the other hand, failing to answer will introduce the stress of losing. The student will have only a limited amount of "hearts," and each wrong answer will remove one. When the player's heart count reaches zero, he loses the game.

3.1.3. Winnable

The game needs to be winnable, but winning cannot be guaranteed, and it should be challenging to finish the game successfully. By selecting a part of the questions, we can select "mandatory" questions that need to be answered to unlock the game finish. After all obligatory questions are answered, the exit needs to be found by a player to win the game. The objectives and exit will be hidden across the whole game board, and the player will collect them to answer a question.

3.1.4. Evaluation method

Evaluation method is needed to ensure that an attempt can be classified as good or bad. A simple scoring system helps to pick the winner among a group of players and enforces the competitive setup. There are multiple statistics we can pick from evaluate the attempt. For example, the student can be evaluated by:

1. the correctness of his answers,
2. the difficulty of the questions,
3. the time spent in the game.

Each correct answer will result in a reward consisting of player in-game experience and currency dependent on the question difficulty. Each wrong answer removes a heart point and gives no reward. In-game currency can be used to buy skills. Therefore we can calculate player punctation by the amount of hearts, coins, and skills.

3.1.5. Environment

The environment requirement is vague and crucial, therefore more in-depth decomposition is required to specify more details.

Following the theme of "generated levels," "objectives," "hearts," "coins," or "experience" we can use the role-playing genre and follow the idea of the student playing a role in some fantasy scenery trying to escape from a maze or dungeon. The inspiration for dungeon environment came from a games of a roguelike type [7], which is a subgenre of role-playing games characterized by dungeon crawling through procedurally generated levels. Many roguelike games also offer skills or items, collected or bought by in-game currency, a limited number of hearts, hit points, some progress counter like experience and level system, or a checklist of tasks to perform while playing. All those elements engage players more into the game world and allow for immersion.

The idea for a maze was to start in the middle of a board and then branch out with multiple paths in many directions. In this approach, if the player finds the exit but is missing any question, he can easily traverse back to the maze center and pick a different path. Furthermore, to make the maze center easy to find and reduce the feeling of being lost, we

decided to place the maze in a circle shape and extend the size of the starting area to make it more distinctive.

For more challenging gameplay, we can significantly reduce the field of view of a player, and the lighting condition of a dungeon will not allow for easy traversing of a maze. This change would require a player to remember visited paths and force him to develop a pattern not to get lost. Reducing the visibility or movement speed also introduces skills to the game, allowing for upgrading selected by a player property. Adding abilities to the game would also fit the roguelike fantasy and create strategies of unique playstyles.

3.1.6. Generic

A mechanic that allows for generic knowledge like in Quizlet Gravity, but with ease of adding own content and managing students' progress like in test platforms like Moodle. For the generic content, a question would be the best. We can display any type of question with any amount of answers. The teacher quickly adds the question to reuse questions from the test.

3.1.7. Replayable

Each student's attempt will result in a new game board with an altered layout and objectives placement, thanks to the randomness. Furthermore, each new session should also be connected to a different scenario with different questions allowing uniqueness and replayability. Game mechanics, like "level" or "experience," could also be persistent across all student's sessions and create some rank that can be used as encouragement and competition mechanic. Most active and best-answering students will achieve a higher level and can be rewarded by a teacher or a parent engaging him even further.

All of the requirements were decomposed and proposed a solution. Having a base for the experience design is essential before moving to implementation.

3.2. MAZE GENERATION

Creating a maze that fulfills the requirements introduced in the previous section 3.1 demands the selection of a proper generation algorithm. Procedural generation allows for creating an almost unlimited number of unique mazes but creates the challenge of significantly reducing the user experience compared to hand-crafted levels. Popular games usually combine hand-crafted elements with procedural generation to take positives from both methods while avoiding the inorganic feeling of the game. First, one-by-one pre-generated cells are used to create an entire board of the circle shape. Next, use a perfect maze generation algorithm to create multiple paths branching from the center.

There are many maze generation algorithms to choose from, and specifying a few parameters by which they can be judged is crucial[2]. We can distinguish:

— implementation difficulty,
— memory usage,
— generation time,
— percentage of dead-ends, straight-ways, junctions, and crossroads,
— possible bias.

More parameters can be proposed, but only these are easily comparable between different algorithms and significantly impact the game feeling.

The usual scenario size will be between ten and forty questions. We can expect the whole game board to have a diameter of not more extensive than thirty. With that knowledge, the memory usage parameter could be skipped. Also, implementation difficulty is not that important because maze generation will be one of the essential elements of the whole game, so there should be no compromises done in this way. For the generation time parameter, the maze should be generated not longer than a second, but with the proposed size of the maze, all algorithms fulfill that so it could also be skipped in the analysis. For the frame of comparison, the measurements in table 3.1, a maze of size 20x20 was used.

Algorithm	Percentage of dead-ends	Bias
Recursive Backtracker	10.19	N/A
Kruskal's Algorithm	30.20	N/A
Prim's Algorithm	34.91	N/A
Aldous-Broder Algorithm	29.00	N/A
Wilson's Algorithm	29.08	N/A
Hunt and Kill Algorithm	10.34	N/A
Growing Tree Algorithm	16.49	N/A
Eller's Algorithm	29.57	N/A
Recursive Division	23.41	Creates patterns
Binary Tree Maze	25.07	Heavy diagonal bias
Sidewinder Maze	27.36	North-facing bias

Table 3.1. Comparison of Maze generation algorithms with the help of Daedalus software [17]

One of the **crucial to avoid is possible bias**. If the maze is biased or repetitive, it will not be attractive. The following two characteristics, percentage of dead-end and straightways, are partially connected because cells can not form straightway if the maze has many dead-ends. This property is not always accurate as algorithms that introduce bias usually have many dead ends and straightways. However, if we consider only algorithms without bias, it is more visible. From those two, the one to be **avoided the most is the number of dead ends**. If the maze has a lot of dead-end cells, it will feel empty, and there will not be enough questions to cover them.

Comparing the properties, the algorithms Recursive Division, Binary Tree Maze, and Sidewinder Maze introduced bias or patterns, so those were excluded[18]. Next, the Prim's, Aldous-Broder, Kruskal's, Wilson's, and Eller's Algorithms were rejected because of many

dead ends that those were generating. So we are left with three options out of the preselected ones: Growing Tree, Recursive Backtracker, and Hunt and Kill Algorithm. All of those algorithms are similar, but Growing Tree statistics for created dead-end and crossroads are a bit worse than the other two. So after rejecting it, only two to choose from are left. The main difference is how they select the next unvisited cell after reaching a dead end. The Hunt and Kill algorithm requires less memory and avoids potential stack overflow that was a problem in Recursive Backtracker, so I decided to use the **Hunt and Kill algorithm** to generate the maze.

3.3. QUESTION TRIGGER PLACEMENT

Another essential element of the game is the placement of the question triggers. Finding those triggers is the session's foremost game mechanic and goal, so finding them must be enjoyable and rewarding.

Placing objects in a maze uniformly random is the simplest but the worst solution. There is no connection between finding a maze branch end and finding an objective.

The second approach could be placing objects in a specific straight line distance from the center. That way, we would place objectives in potentially better places than the entirely random place, but still not ideal as objects may stack one-by-one in a long outer corridor. Also, the distance from the center does not equal the actual distance that the player needs to travel.

The third option is to count the actual distance to the starting cell instead of the straight line distance. That way, we would know how deeply the objective is hidden from a player. However, one big downside of this solution is that the longest maze path will probably contain many or even the largest distances, which means that all objects will be spawned in one corridor.

The last and optimal solution would be to use the property of the cell being a dead-end. When generating the maze, we could mark dead-ends cells and use this knowledge to fill them with objectives. Thanks to that, we could make the maze less empty and, at the same time, reach branch end more rewarding for a player.

4. IMPLEMENTATION

For the implementation part, we can distinguish two elements:

— implementation of the whole solution as a team effort,
— implementation of an educational video game as an individual work.

This chapter will also describe the networking part used to ensure the game module and server application connection.

4.1. ENTIRE SOLUTION

The solution contains six distinctive elements, with each providing a critical functionality:

— backend ASP.NET application for handling communication between modules,
— game created with Unity, available for students, connecting to the backend application,
— AI module, evaluating question difficulty and creating new questions based on textbooks and previous questions,
— database solution, PostgreSQL storing question, users, and prior sessions
— adaptive module, backend module responsible for adapting question difficulty to the student level
— Web management system with Vue.js, available for teachers to create classes, add new students, create and schedule scenarios, browse student results.

The game is only one of the elements of the entire solution created as a team effort, but the main focus of this work is the game, so this is only a brief introduction to its implementation and now the focus will be centered on the game implementation.

4.2. GAME ENGINE

Unity[16] game engine fulfilled all the requirements introduced, so it was selected. It is also the most popular and accessible for use in small projects. Furthermore, the created game used a 3D world view with the 2D user interface and was deployed primarily as a web game, so those were also essential requirements for choosing the game engine. Finally, it was also well known to me to shorten the development significantly.

4.3. GAME PLATFORM

WebGL[1] was selected as the leading platform because of its ease of use and availability. It does not require installation and is compatible with almost every browser. Students or schools do not need to have access to a high-performance device. WebGL provides excellent graphics and user experience while having better performance than flash games.

While having many advantages, it also has drawbacks, like it is slower than system native apps or OpenGL. WebGL also introduces some problems with the networking and connection of the game to the server. We used the Unity cross-platform ability to create Android and iOS builds for students who have no access to non-mobile devices to solve those problems.

4.4. THE CONNECTION BETWEEN GAME AND SERVER

Because the game can be hosted on a completely different hosting than the server and is modular, the networking connection layer between them is required. We decided to use REST API with Google's Protobuf [6].

The first solution was based on gRPC with Protobuf, but we discovered that WebGL is incompatible with gRPC. To solve this problem, instead of the gRPC, we used the RESTful approach with HTTPS and endpoints for each needed request. The game uses the ConnectionManager class and UnityWebRequest to send requests and receive responses. Protobuf allows for the same data model on the game frontend and server backend faster than, i.e., JSON [15]

4.5. HUNT AND KILL ALGORITHM

To generate the maze selected in 3.2, Hunt and Kill[4] algorithm was used. The significant advantage of using it is that we can select any cell inside the maze as a starting cell. It is essential as the game will be using the board with starting position in the center. Hunt and Kill algorithm can be described easily with the list of steps given below.

1. Mark all cells not-visited.
2. Choose starting cell mark it visited. This is the current cell.
3. Pick a cell adjacent to the current one that has not been visited: visit it and mark it as the current one.
4. Repeat point 2. until no adjacent cell can be selected - **The Kill**.
5. The previous cell becomes the current cell. If this cell is the starting cell, then we are done. Else go to point 2. - **The Hunt**.

The algorithm can be split into two parts, first the "Kill" phase, where we go to the next adjacent unvisited cell until there are no more cells. Then, we execute the second phase,

"Hunt," where we search for an unvisited adjacent cell of any visited cell. Finally, we go back to the "Kill" phase if we find one. If not, the maze is ready. The loop can be easily explained with the help of a picture of the board state after one algorithm cycle. In picture 4.1, all cells start as white: this is equivalent of marking them not-visited in point 1 of the algorithm. Point 2, represented by the green cell, is the starting point we selected. From starting cell, we repeat step 3 until we reach a dead end in point 4. The last part of the algorithm in point 5 selects the Hunt cell, marked with blue color. Now we start repeating from step 3 until all cells are visited.

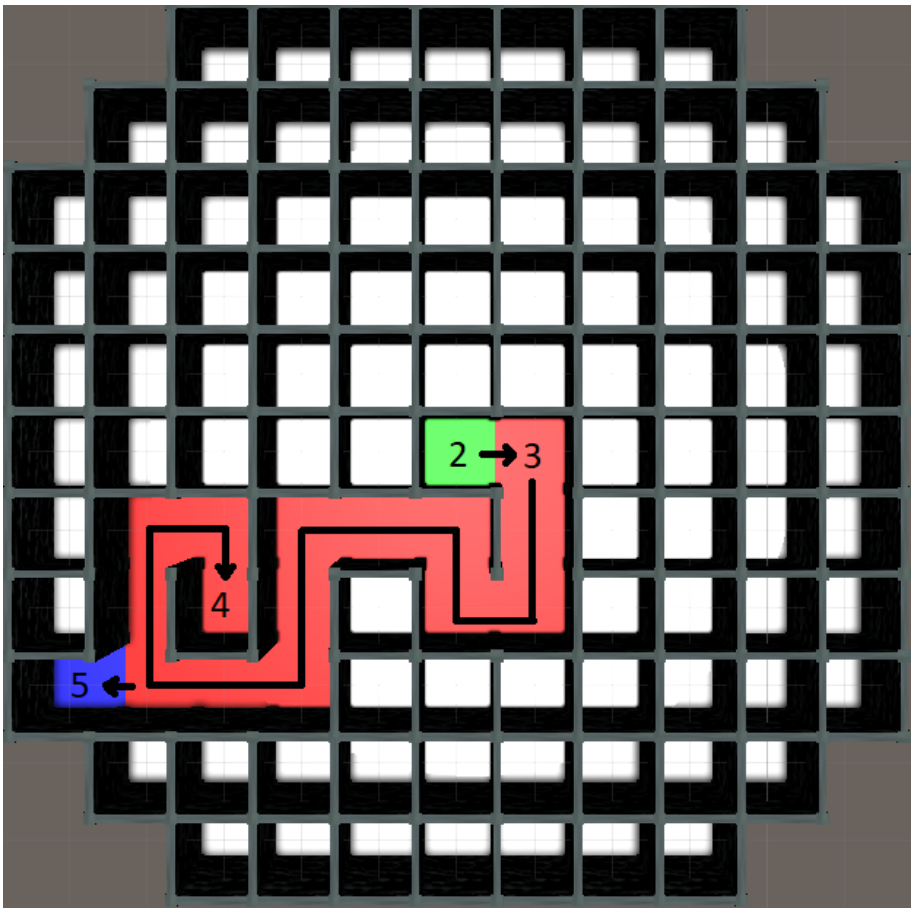

Fig. 4.1. Hunt and kill maze generation cycle

This algorithm also has a few possible alterations and optimizations. One important feature is: how we hunt. In the basic implementation, we iterate the whole board from coordinated (0, 0) to (n, n) so the maze will have more new parts going from the top left corner. We can also trace back from the last dead end until we find a new unvisited cell, but this solution will favor long corridors instead. The implementation used in the game is to hunt cells from the center toward the outer part of the board. This way, we promote longer branches instead of some bias or one long branch.

Another extension to this algorithm is to use procedural generation keys instead of always using a new random direction. For example, at the start of maze generation, a key

of the length of the board diameter is generated with four possible directions. Then, we iterate through this key when selecting a new direction for a "Kill" instead of randomly choosing a direction. That allows for a more "organic" feeling because of a repetitive flow instead of pure randomness.

All those extensions allow for the natural feeling of the maze and improve the user experience when traversing it.

4.6. QUESTION TRIGGER PLACEMENT

An essential part of the game board generation is to place the objectives in a rewarding way as the previously discussed optimal solution is placing them in dead-ends of the maze. To implement that, extension to Hunt and Kill algorithm was added that marks each cell's actual distance from the center and marks all the dead-ends. After the maze is generated, it is time to place all the objectives. First, a list of dead-ends is collected shuffled. Then we check if there are enough cells to place all objectives. If not, we also append random cells with distance from the center larger than board diameter to the list. After we have a list of cells, we can spawn trigger objects in the maze, filling all the dead ends with a "reward" for a player.

One specific type of question trigger is the "enemy" trigger. This object is not stationary, but he is also traversing the maze while the player is. There is no need to place this type inside dead ends as he is moving and at the player reaches this dead-end, he will be already somewhere else. This type of objective adds some dynamics to the game as the player will see movement and either try to run away from them or try to catch them.

4.7. USER INTERFACE AND ART STYLE IMPLEMENTATION

The covered above implementation is only responsible for the game board generation. Making a complete and attractive game is not enough to create a board, player, and objectives. The game also contains:

— menu with a login panel,
— the tutorial system, explaining game basics,
— the User Interface system with:
    1. head-up display (HUD) with all statistics, skills, objectives, or buttons,
    2. windows for questions, question result, game win or loss, pause,
    3. popups with tips for the player,
— art style with animations and particle systems to make the game visually attractive.

The last part of the implementation was the creation of an offline mode for the game. This mode does not require logging in or internet connection but is limited with only a few questions with no adaptability and no tracking process. This mode was created mainly for

more accessible and quicker user experience testing without signing with accurate data. In addition, the testers' anonymity should allow for more objective opinions on the game.

5. RESULTS

The output of this work can be easily presented as a working game, but not only the generated game is essential. The user testing and users' opinion about the game are even more crucial.

5.1. THE DEVELOPED GAME

The output of this work can be easily presented as a working game. In picture 5.1, the game view on the board is presented, while in picture 5.2, the question window can be seen.

Fig. 5.1. Game core loop screenshot. *The maze exit is visible in the right left corner, the player character in the center, and the "enemy" question trigger on the player's right. Also, the Head-over display User Interface is visible on this screen.*

5.2. USER TESTING

A series of tests were performed across the whole development phase. In the game development industry, it is essential to always ask for feedback from the end-users during

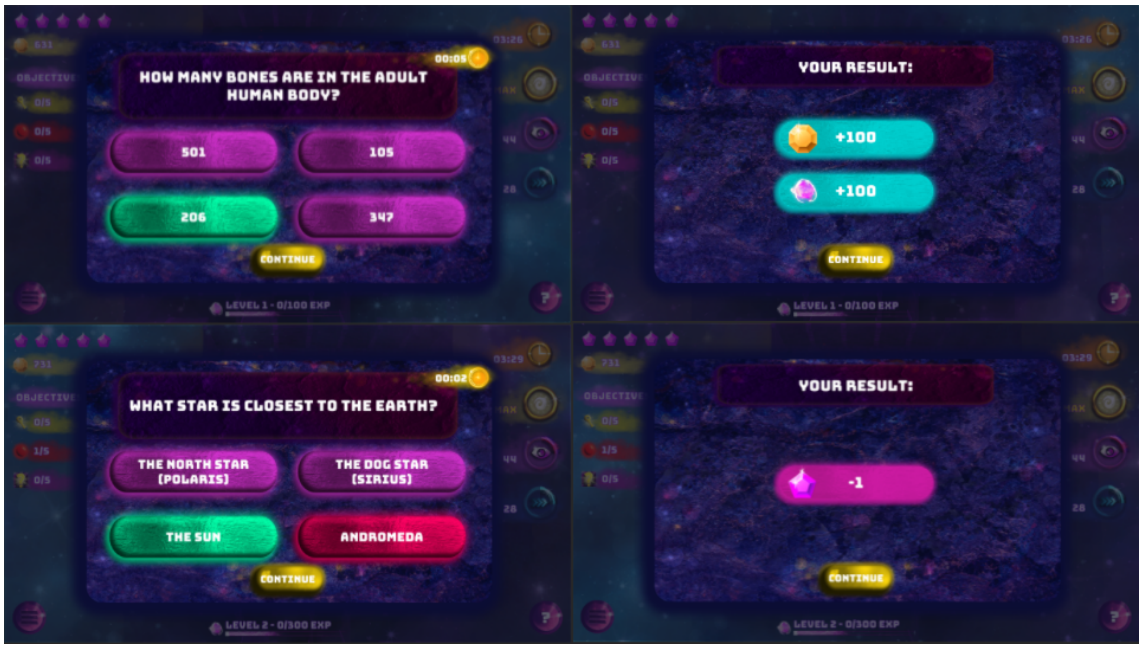

Fig. 5.2. Question window screenshot, *correctly answered question (left-up corner), the reward for correct question (right-up corner), wrongly answered the question (left-down corner), the result for wrong answer (right-down corner)*

the development. Therefore, standard testing methods conduct a feedback session with potential users in an interview or feedback survey.

5.2.1. Feedback and user experience

During the early stages of development, user interviews were performed to ensure the game implementation was going in the right direction. Whenever there were many potential directions or any uncertainty, multiple choices were presented to a potential end-user to help decide which option fit the needs. For example, this method was used when deciding on objective types, skills selection, or art style. Users were presented with multiple options and were asked to select one they would like to see in a game and why.

As a second method of gathering users' opinions, we prepared an **online survey with eight line-scale questions and optional comments** about contact and age group. It was created and sent to all users that tested the game. Questions included topics:

— art style,
— gameplay, *a summary on figure 5.3*,
— maze generation, *a summary on figure 5.4*,
— user Interface,
— skills,
— questions,
— game idea,

— performance.

While for this thesis, the most important is Gameplay and Maze generation. Other aspects of the game also influence the quality of the experience and user enjoyment, so it would be hard to judge the solution without a complete picture.

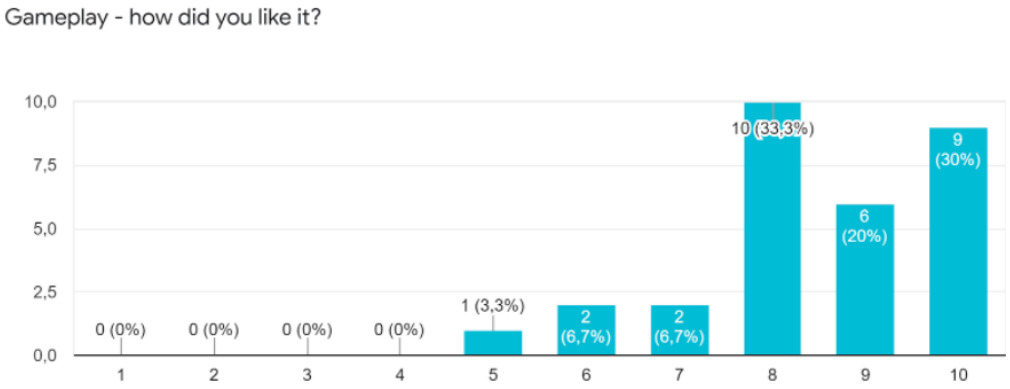

Fig. 5.3. Gameplay experience feedback

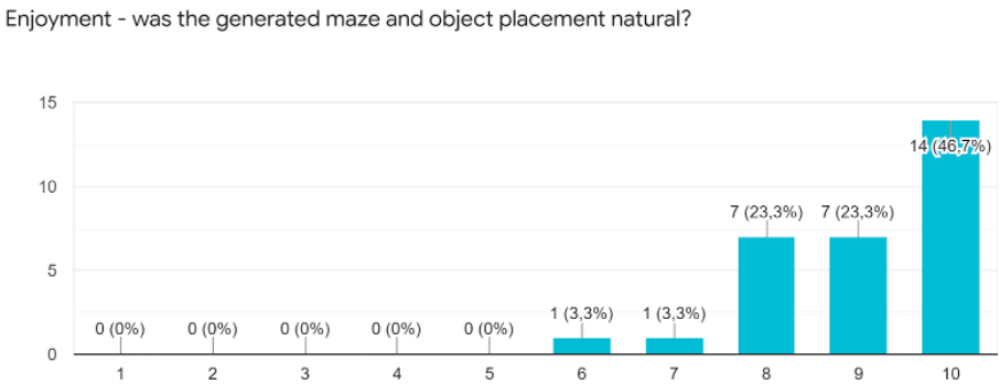

Fig. 5.4. Game board enjoyment feedback

Because of limited time and project budget, only around a hundred players played the game, and many of them were not from our target age group. In addition, the game is working in online and offline mode and while tracking online games is easy, it is way harder to track offline players. From the survey, we received thirty answers.

There are not many text answers, but general opinion could be formed based on them. Some positive feedback that indicates the attractiveness and enjoyment requirements were met.

— It was quite nice, really was thinking that maze was done manually.
— Did not run into any problems, it felt natural [...]
— The maze was not obvious to go through and in general should entertain puzzle-lovers [...]

— I liked the gameplay of searching for the keys [...]
— it's fun to run away from the red dots [...]
— The game is not boring.
— This maze was easy with moving[...]
— It was pleasant[...]
— The UI is clear and understandable. The FTUE and all the popups communicate well what they should[...]
— The visuals are simple but they are clear and readable[...]
— Great way to learn[...]
— I really liked the idea, I think it is a good way of expanding your knowledge[...]

But also indicate some room for improvements.

— I was a little bored after finding few keys [...]
— younger player might not have an easy task getting out [...]
— decorations or special rooms could improve the monotony [...]
— Sometimes the objectives in the maze get clustered in one place [...]
— I don't understood some of them [UI elements] [...]
— Did not understand what the first power [skill] did [...]
— I missed the skills after I started concentrating on finding the keys.
— brightness [skill] not much changing anything or speed, and its a bit not fair that price is rising so much [...]
— more clear whether this is a multichoice question [...]
— try to differentiate between objectives [...]

In summary the game was enjoyable for players, the basing goal of the game was understandable and most of people found it attractive.

5.2.2. Improvements after feedback

Collecting feedback is essential in game development, but the changes applied to the solution after the feedback are even more critical—examples of implemented modifications are mostly connected to the first-time user experience and user experience in general. For instance, players had problems using skills or did not understand the goal and objectives of the game. While there was already an existing tutorial, it was not enough.

To address the problem that users were not noticing skills buttons or forgetting about them, different skill buttons graphics were selected, and animations of clicking were added. Furthermore, animated arrows highlight skill buttons when players have enough in-game currency to purchase them.

In addition, understanding the game goal was improved by adding an objectives panel that is always visible on the screen and text tips that appear when possible action is available. i.e., after selecting the last mandatory objective, a recommendation to search for the exit

will appear, or when the player tries to exit without completing the goal, a tip will appear explaining what he has to do now.

To summarize, the improvements made possible thanks to players feedback helped with:

— changed the color palette of the game,
— changed the skills icons and animations
— improved the accessibility by adding joystick on top of keyboard player control,
— added tips reminding player the goal of the game,
— improved the exit visibility,

and other more minor changes.

6. CONCLUSIONS

The project was successful, attracting many players and leaving mostly positive opinions. Selected game setup and appropriate generation algorithms made the game engaging and enjoyable. My contribution to this project is:

— designing the user experience,
— implementing a maze generation and object placement algorithm,
— implemented common video mechanics and user interface,
— playtesting, interviews with players and questionnaires for analysis player experience evaluation,
— connection of the game with the server.

In general, my contribution was everything connected to game design and testing.

6.1. FURTHER DIRECTIONS

While the solutions present promising results, there is much potential for future improvements and directions. The base game servers only as a proof of concept, and there is a lot of new mechanics, features that would further extend the user engagement. The improvements for the current version:

— balance and more explanation for skills,
— adding new question triggers types,
— adding new skills,

In contrast, adding new mechanics that are not yet present could also improve the attractiveness of the game. Features like a player leaderboard or change in character or board appearance could improve players' retention and engagement. A new game loop with "saga" mode where students could play on demand and not on teacher invitations would provide more game content for active players. The management panel could also be made available for parents, not only for teachers. A helpful extension would also be adding more statistics from the student session to understand better.

BIBLIOGRAPHY

[1] Khronos WebGL Working Group, *Web graphics library*, https://www.khronos.org/webgl/. [Online]. Accessed 2021-12-14.

[2] Bellot, V., Cautrès, M., Favreau, J.M., Gonzalez-Thauvin, M., Lafourcade, P., Le Cornec, K., Mosnier, B., Rivière-Wekstein, S., *How to Generate Perfect Mazes?*, Information Sciences. 2021.

[3] Bernhaupt, R., *Evaluating User Experience in Games: Concepts and Methods* (2010).

[4] Buck, J., *Mazes for Programmers: Code Your Own Twisty Little Passages*, 1 wyd. (Pragmatic Bookshelf, June 2015).

[5] Conscious Content Media, Inc., *Homer game*, https://learnwithhomer.com. [Online]. Accessed 2021-12-11.

[6] Google, *Protocol buffers*, https://developers.google.com/protocol-buffers/docs/overview. [Online]. Accessed 2021-12-13.

[7] Harris, J., *Exploring Roguelike Games Book*, 1st edition (september 10, 2020) wyd. (CRC Press, 2020).

[8] Informa Tech, G., *What is games 'user experience' (ux) and how does it help?*, https://www.gamedeveloper.com/business/what-is-games-user-experience-ux-and-how-does-it-help-. [Online].

[9] Lee, J., Hammer, J., *Gamification in education: What, how, why bother?*, Academic Exchange Quarterly. 2011, tom 15, str. 1–5.

[10] Microsoft, *Minecraft education edition*, https://education.minecraft.net/en-us/homepage. [Online]. Accessed 2021-12-11.

[11] Nintendo, *Big brain academy*, https://www.nintendo.com/games/detail/big-brain-academy-brain-vs-brain-switch/. [Online]. Accessed 2021-12-11.

[12] Pivec, M., Kearney, P., *Games for learning and learning from games*, Informatica (Slovenia). 2007, tom 31, str. 419–423.

[13] Prodigy Education Inc., *Prodigy game*, https://www.prodigygame.com/main-en/. [Online]. Accessed 2021-12-11.

[14] Quizlet Inc., *Quizlet gravity*, https://quizlet.com/110201463/gravity. [Online]. Accessed 2021-12-11.

[15] Sumaray, A., Makki, S.K., *A comparison of data serialization formats for optimal efficiency on a mobile platform*, w: *Proceedings of the 6th international conference on ubiquitous information management and communication* (2012), str. 1–6.

[16] Unity Technologies, *Unity*, https://unity.com/releases/2020-1. [Online]. Accessed 2021-12-14.

[17] Walter D. Pullen, *Daedalus [computer software]*, http://www.astrolog.org/labyrnth/daedalus.htm. [Online]. Accessed 2021-12-04.

[18] Walter D. Pullen, *Maze classification*, http://www.astrolog.org/labyrnth/algrithm.htm. [Online]. Accessed 2021-12-04.

[19] Zapalska, A., Brozik, D., Rudd, D., *Development of active learning with simulations and games.*, Online Submission. 2012.