VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Tower Defense. 2D game with strategy and AI features.

Done by:
Oleh Petrov
Nojus Vislabokas

Supervisor:
dr. Linas Būtėnas

Vilnius
2023

Preface

This project was done during the 3rd semester of the study programme *Information Technologies* in the specialization of Innovative studies. We have chosen the topic *Tower Defense. 2D game with strategy and AI features.* suggested during the project market on the first lecture of the subject "Problem-Based Project". The topic seemed very interesting.

January 12, 2023

Oleh Petrov

Nojus Vislabokas

Contents

Abstract

The poster presents "Monsters and Mages" - fantasy themed 2D tower defense game. Tower defense is a popular game genre of real-time strategy which requires difficulty balancing and content for replayability to make a fun player experience. First the famous “Bloons TD” games were analyzed for inspiration on gamemodes, enemies and game mechanics. A game development engine "Unity" will be used to make the 2D Tower defense game using C# Language for Windows platforms. The Game will have PostgreSQL database to store player progress and a webserver to connect this database with the game client. The expected result is to have a player vs AI game where enemies walk in a given path and player has to use currency to defend against a horde of enemy waves. There will be playable maps with many towers and enemy variations. You will be allowed to pick a "Challenge" gamemode where you will be placed in a map with conditions, where you can only use certain towers. Also a "Sandbox" gamemode where you will be allowed to place as many towers as you want and and fight a ever increasing difficulty of monsters, just like in the "Endless" gamemode, where you need to hold out as long as you can. There will be boss monster that have special abilities like other, more weak monsters. Users will be able to save data by creating an account and transferring data by logging in into another device. So player don’t need to worry about losing their data.

Keywords: Tower Defense, Game, Unity, C#, 2D

Santrauka

Bokštu gynybos žaidimas

Plakate pristatomas „Monsters and Mages“ – fantazijos tematikos 2D bokšto gynybos žaidimas. Bokštų gynyba yra populiarus žaidimo žanras, skirtas realaus laiko strategijai, kuriam reikia sunkumo balansavimo ir pagaminti toki turinį, kad būtų galima pakartotinai žaisti, kad žaidėjui būtu linksma. Pirmiausia garsieji „Bloons TD“ žaidimai buvo išanalizuoti, ieškant įkvėpimo apie žaidimo režimus, priešus ir žaidimų mechaniką. Žaidimu kurimo variklis pavadinimu "Unity" bus naudojamas kuriant 2D bokštų gynybos žaidimą naudojant C# kalbą, skirtą Windows platformoms. Žaidimas turės „PostgreSQL“ duomenų bazę, kurioje bus saugoma žaidėjo pažanga, ir tinklapio serverį, kad tai būtų galima sujungti duomenų bazę su žaidimo klientu. Tikimasi, kad žaidimas bus žaidėjas prieš AI, kur priešai eis nurodytu keliu ir žaidėjas turi naudoti gautus pinigus, kad apsigintų nuo priešu bangų minios. Bus žaidžiami žemėlapiai su daugybe bokštų ir priešu variantų. Bus galima dar rinktis "Challenge" režimą, kuriame jūs busite patalpintas į žemėlapyje su sąlygomis, kur galėsite naudotis tik tam tikrais bokštais. Tuo pačiu dar "Sandbox" žaidimo režimas, kuriame jums bus leista pastatyti tiek bokštų, kiek norite, ir kovoti prieš vis sunkėjančius priešu bangas, tai pat kaip ir "Endless" režime, kur reikia išsilaikyti kiek tik išeina. Bus viršininkai monstrai, turintys ypatingų sugebėjimų, kaip ir kiti, silpnesni monstrai. Vartotojai galės išsaugoti duomenis paskyros kūrime ir duomenų perkėlimą prisijungus prie kito įrenginio. Taigi žaidėjui nereikia nerimauti dėl duomenų praradimo.

Raktiniai žodžiai: Bokštu gynyba, žaidimas, Unity, C#, 2D

Introduction

The video game is a top-down 2D Tower Defense game. Made for Windows desktop devices. The theme of our game is magic. The enemies of the player are fantasy monsters, whereas the towers are magical wizards. The Video game consists of many waves, the player loses the game if the monsters reach the end of the level, as one monster damages a specified amount of Health Points and if it reaches 0 the game ends. The enemies and wizards all have special abilities that either help the player or do not.

1 Preliminary design

This section discusses the preliminary design (figure 1) of the application concerning the game interactability. The design shown in figure 1 is a representation of the interactivity that is supposed to be provided for the user.

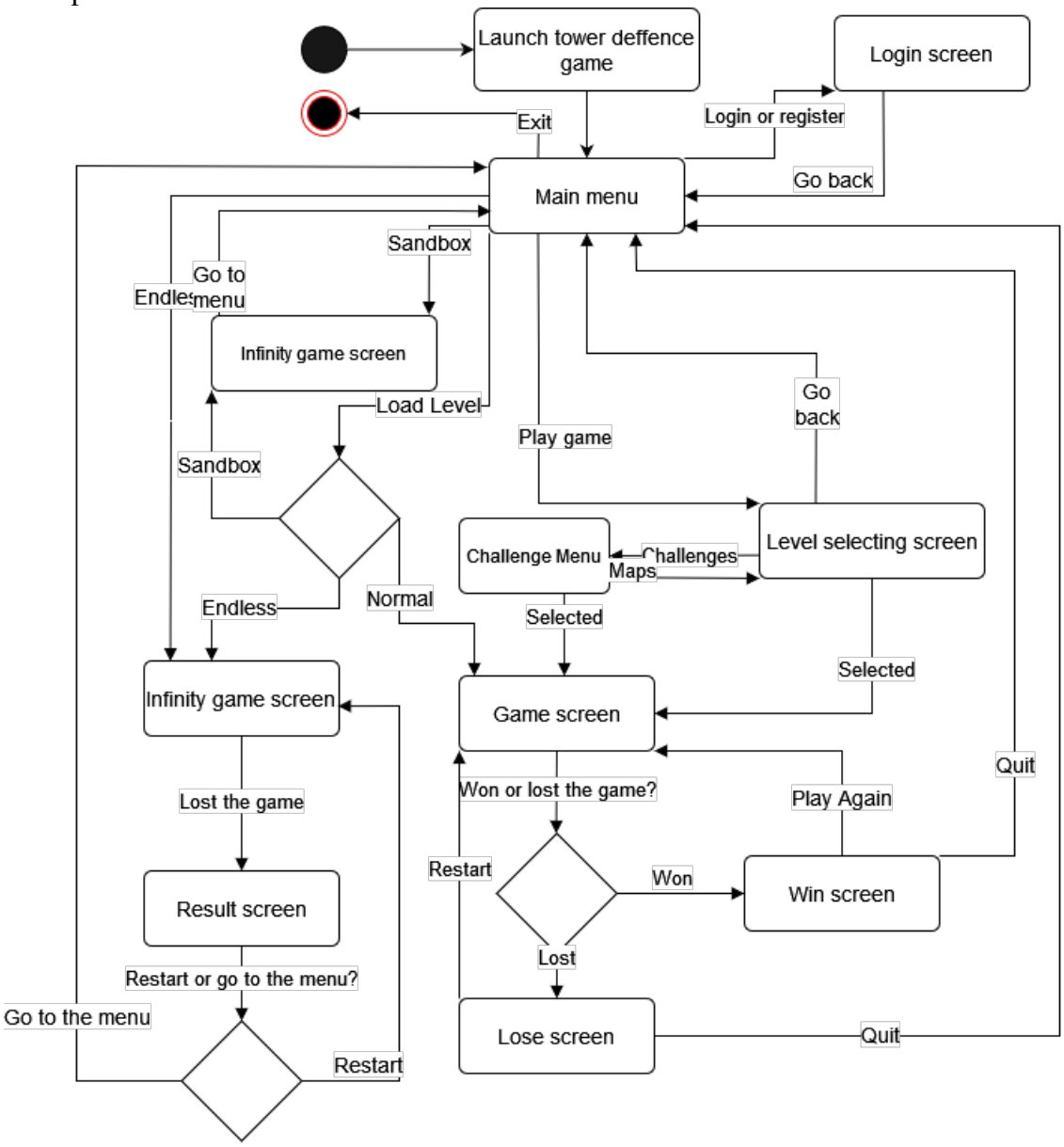

Figure 1. Preliminary design wireframe of the Tower Defense game application

1.1 Menu

Upon opening the application, the user shall be presented with the main menu. The user will be presented with several choices, including “Play”, “Sandbox”, “Endless” and “Exit”, while also being able to login in or register as a user. The option to login lets the user save their data of beaten levels so when they download the application on another computer, their progress isn’t lost. The "Sandbox" and "Endless" choices lets the play in an endless level that gets difficult bit by bit, the difference being that in the "Sandbox" choice you are allowed to place and tryout towers for free without the concern for loosing, while the "Endless" option throws you in a hardcore gameplay enviroment. The "Play" option opens up the map menu where you can choose what map you want to play or go to a challenge menu, where you can play levels with set conditions.
Passing a round in a level saves the current state of the map and adds a new choice in the main menu called "Load Game", which allows the player to join back to the progress he has made while playing.

1.2 Game screen

When you go into a level, you are faced with your map, your health that will dictate how close you are to loosing and your currency that you will use to place your towers. The light brown tiles are sand they are used as the path the enemies will walk on and the green tiles are grass tiles you can place your towers on. You are given 5 towers: the cheapest and basic tower - the fire wizard that shoots normaly and damages enemies, a long distance attacking tower - the electro wizard, able to shoot very far but not close to himself, the explosive area of effect tower - the scientist wizard, able to shoot and damage many enemies that are close to each other, a tower that slows enemies down - the goo wizard, who shoots out goo and puts a slowness debuff on the enemies and lastly a close range attacking tower - the ice wizard, who attack enemies stading around him. You are able to place these towers and use them to defend against the waves of enemies, but your also are given the option to sell these towers if there is a need for it. After pressing the "Next wave" button to start the next wave, you can fast forward button to speed up the action.

2 System architecture

2.1 High-level Overview

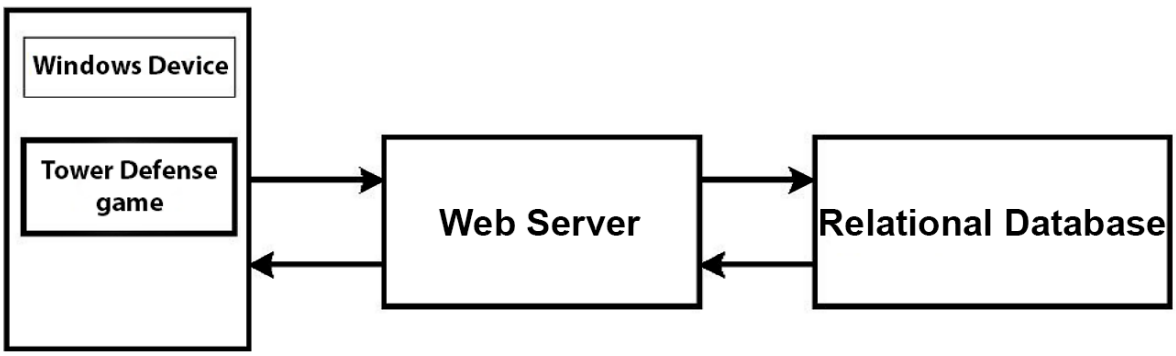

Figure 2. High-level overview of the Tower-Defense game application

The Video Game Application - the video game itself where players will be able to play and interact.
The Database - a relational database that stores player data, which contains the player's progression.
The Web Server - is used to connect the database and the client side Windows app. It its serves HTTP requests by fetching data from the database and sendind it back in JSON format.

2.2 Deployment

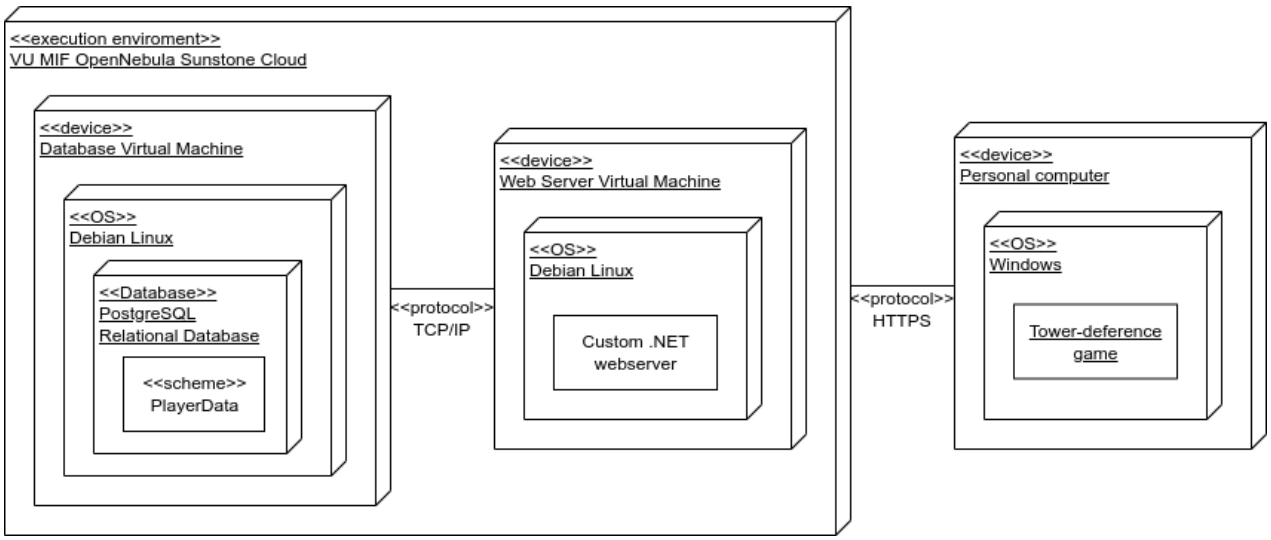

Figure 3. UML Deployment Diagram

UML deployment diagram - overview of the whole system showing the deployment environment and high level interactions between the main components.

2.3 Technologies and Tools

The client-side Windows app is made in Unity Game Engine using the C# programming language. To generate main menu background used DALL-E AI.

The web server is programmed using C# programming language and .Net Core framework, using the JetBrains Rider IDE. To run it, NGINX is being used inside of a Linux virtual machine. It is running a web server, which enables sending and receiving data from the client-side app. The web server securely communicates with the client-side using HTTPS protocol.

For the database part, PostgreSQL relational database management system is used. It is running inside of a Linux virtual machine.

2.4 Front-end

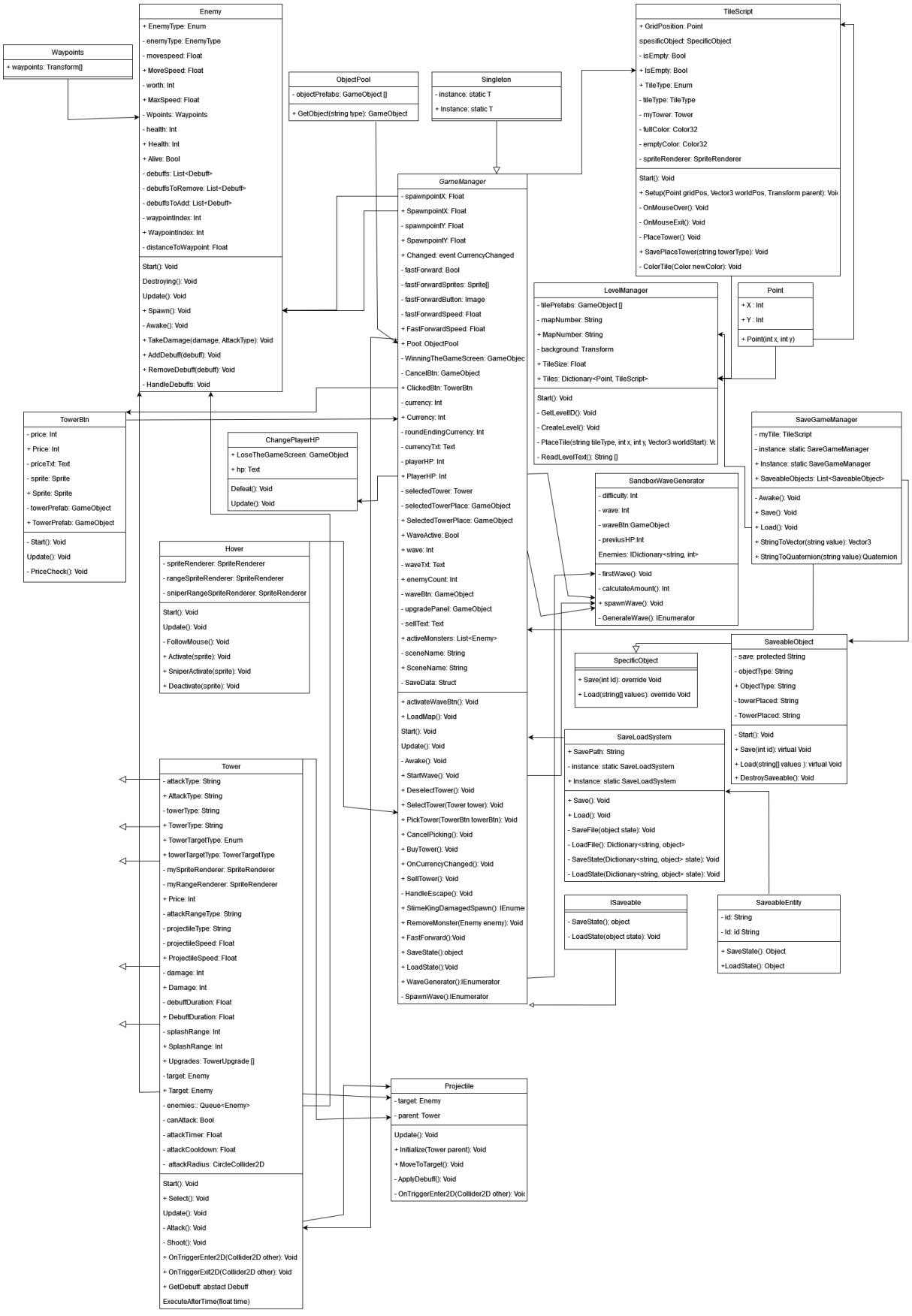

Figure 4. UML Diagram part 1

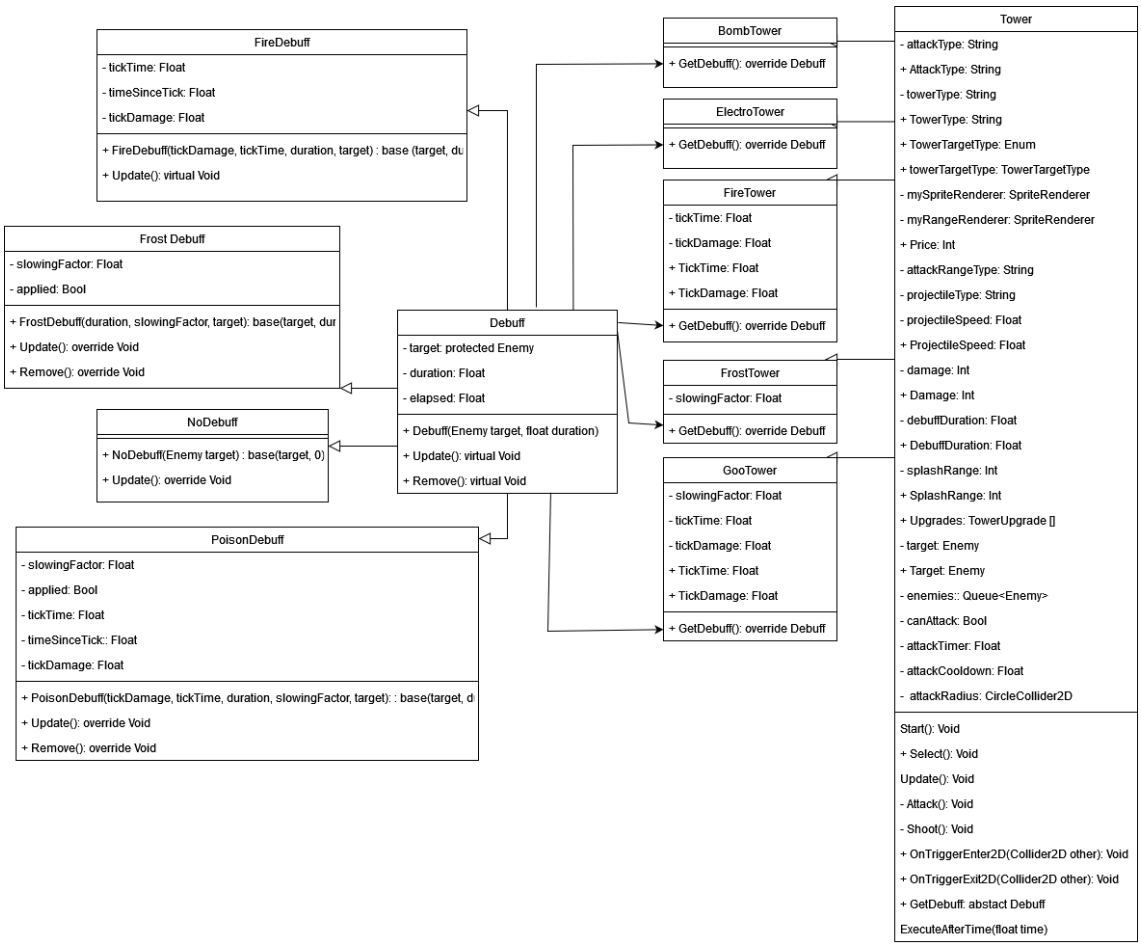

Figure 5. UML Diagram part 2

This UML diagram shows the required variables and functions in the front-end of the game application. This class diagram shows only the functions and variables that are used with the main GameManager script in one way or another. Main menu, Scene changing and web scripts are not shown in this diagram.

2.5 Back-end

The back-end mainly consists of 2 parts: web server and database. Both of them are deployed on OpenNebula on separate virtual machines. They are running natively on Debian.

2.5.1 Overview

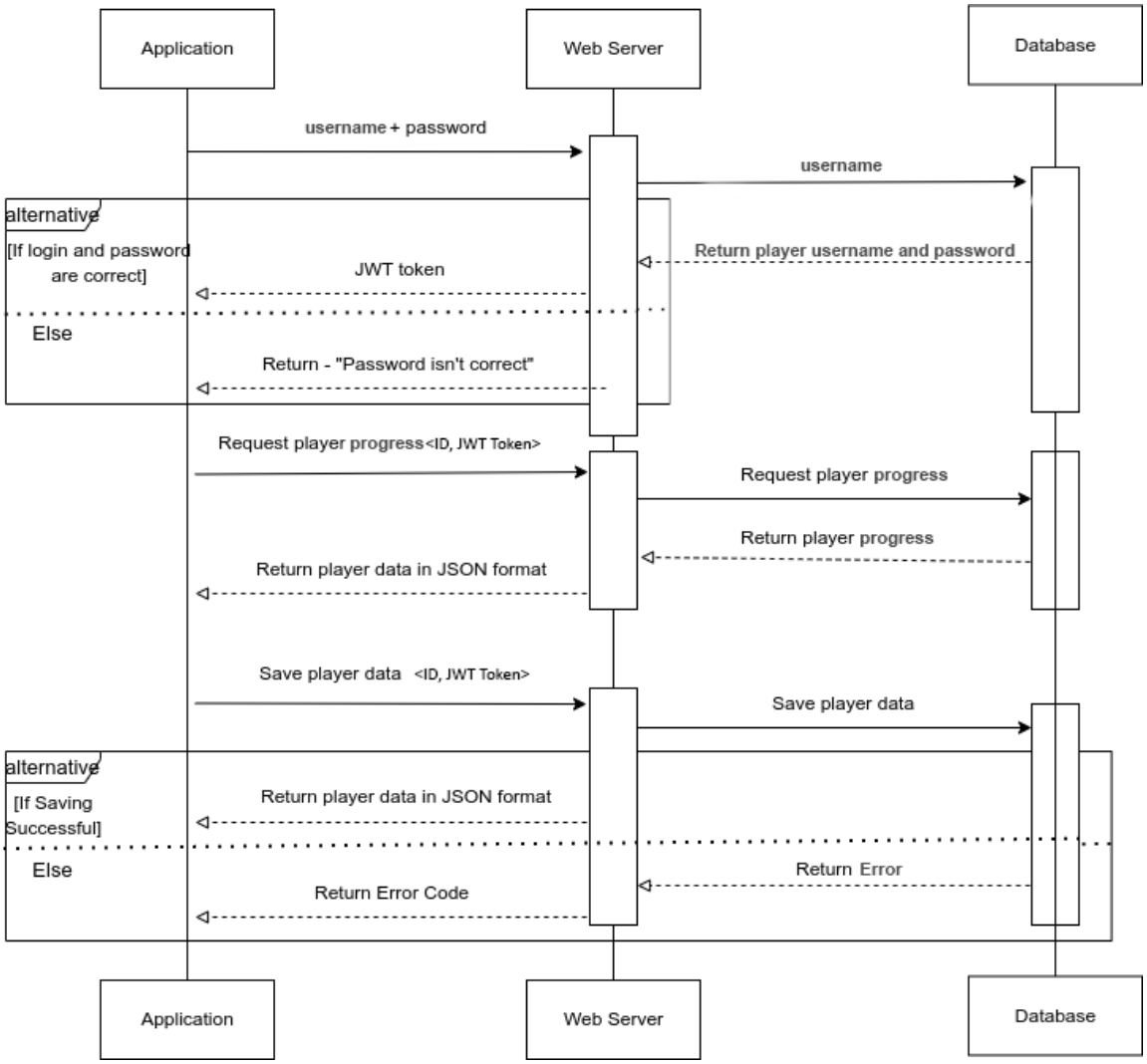

Figure 6. UML sequence diagram

This diagram explains what are the main parts/modules/components and how they all connect together.

2.5.2 Database

This project uses PostgreSQL 12 as relational database. Its schema was sourced from ER model (Fig. 7) To fulfill the fifth point in the description of the ER model, a SQL function was created along with additional restrictions. Together they check the correctness of the transferred game session data by checking the relationships between the tables, the passed map cannot be saved if there is no such card in the map table and if there is no username in the player table.

2.5.3 Web Server

The web server is running on .Net Core framework. To run it, NGINX is being used inside of a Linux virtual machine. Web server enables sending and receiving data from the client-side app. The web server securely communicates with the client-side using HTTPS protocol.

The Web server has all the necessary functions for loading, adding and modifying data in the database.

The web server has the function of generating JWT tokens that will be valid for a week from the moment of generation. Player progress data can only be obtained and modified by having this token.

Unit testing can be done in harmless way because of the usage of interfaces and modular architecture. This means that components can be tested in isolation.to be in unit tests.

2.6 Conceptual Modelling

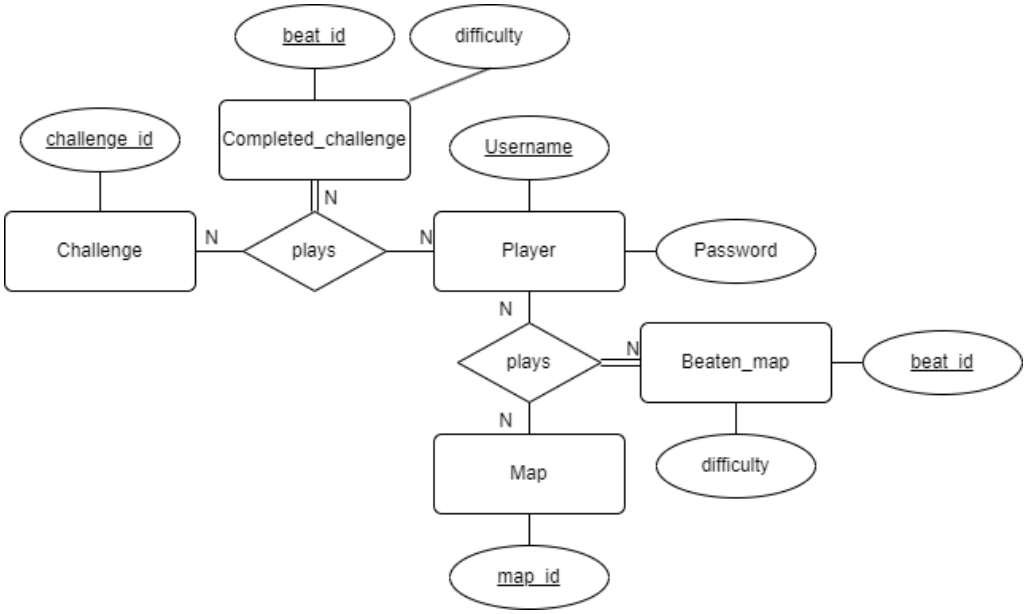

Figure 7. Entity Relation diagram

This diagram (see Figure 7) represents all the relationships in our database. the structure is completely based on it. There are 5 entities: player, challenge, map, passed_challenge, beaten_map. Apart from these, there are several relationships between:

1. Each player can beat any map
2. Each player can beat any challenge
3. A player can't beat a challenge that doesn't exist
4. A player can't beat a map that doesn't exist

The **"player"** entity represents an authorized unique player. It has attributes that allow the player to securely log into their account as well as in-game currency to buy in-game items.

The **"challenge"** entity has an atribute which represent data about available challenges.

The **"map"** entity has an atribute which represent data about available maps.

The **"completed_challenge"** entity has attributes which represent data which challenges player beat, and in which difficulty.

The **"beaten_challenge"** entity has attributes which represent data which maps player beat, and in which difficulty.

3 Functional Requirements

3.1 High priority

- Have manually made waves of enemies, consisting of 20 waves.
- Have placeable towers and a shooting system.
- Have playable maps.

3.2 Medium Priority

- Give towers uniqueness like having the ability to create status effects for enemies, having area of effect damage or different type of ranges.
- A Saving function to save current map progress or beaten map progress.

3.3 Low Priority

- Make a game state save and load function.
- Have a sandbox gamemode where you have infinite of everything, a endless gamemode of infinite waves that get harder and a challenge gamemode where you beat a level with conditions.

4 Non-functional Requirements

4.1 Compatibility

- The game must be able to run on the majority of Windows devices.

4.2 Reliability

- The game should not have any issues or bugs that prevent the player from a non-frustrating experience.

4.3 Security

- The database and the server should be in separate virtual machines due to security reasons.
- Web server using JWT Tokens to autenticate player, therefore, the data from every account cannot be downloaded or received from outside.
- The player's password and login are available only in the database and inside the server, the client receives only a token.

4.4 Performance

- The game should work well on a computer that can run Windows 7,8,10 reliably.

5 Algorithms and Data Storage

5.1 Algorithm Overview

Function Attack
Input:
attackTimer float
attackCooldown float
Output: None

```
1   If canAttack = false Then
2     attackTimer += Time.deltaTime;
3          if attackTimer >= attackCooldown Then
4     canAttack = true;
5     attackTimer = 0;
6
7   if the target = null && enemies.Count > then 0 Then
8     target = enemies.Dequeue();
9   if the target != null Then
10    if tower canAttack Then
11      Shoot();
12      canAttack = false;
```

Figure 8. Pseodo code of function *Attack*

Function Shoot
Input:
attackType string
projectileType string
Output: None

```
1   if the attackType == "Ranged" Then
2     Projectile projectile = GameManager.Instance.Pool.GetObject(projectileType). 3
GetComponent<Projectile>();
4     projectile.transform.position = transform.position;
5     projectile.Initialize(this);
6
7   else if the attackType == "Melee" Then
8     var hitColliders = Physics2D.OverlapCircleAll(transform.position, SplashRange);
9     foreach (var hitCollider in hitColliders)
10        var enemy = hitCollider.GetComponent<Enemy>();
11       if its an enemy Then
12          enemy.TakeDamage(Damage);
```

Figure 9. Pseodo code of function *Shoot*

5.2 Real World Example

Using first set of example data:

- In the Attack function on line 1 in the IF statement THEN section will be skipped because, as default the canAttack is equal to true.
- On line 7 it will check IF the tower doesn't have a target AND IF the queue has more targets inside, if true then it will remove the front queue and make it the new target.
- On line 9 IF the tower has a target then the THEN section will not be skipped .
- On line 10 in the IF statement THEN section will not be skipped and the Shoot function will be called, while changing the canAttack to false.
- In the Shoot function on line 1 IF statement THEN section will not be skipped since the attackType is "Ranged", this will create a projectileType called "FireBolt" and place it in the middle of the tower, with its own script for attacking.
- On line 6 in the IF statement THEN section will be skipped because its attackType is not equal to "Melee".
- On the Attack function on line 1 in the IF statement THEN section will not be skipped because now canAttack is false, so every second the attackTimer integer will grow .
- On line 3 IF statement THEN section will be started when 1.25 seconds have passed for attackTimer to reach attackCooldown to let the tower attack again and reset the attackTimer count.

Using second set of example data:

- In the Attack function on line 1 in the IF statement THEN section will be skipped because as default the canAttack is equal to true.
- On line 7 it will check IF the tower doesn't have a target AND IF the queue has more targets inside, if true then it will remove the front queue and make it the new target.
- On line 9 IF the tower has a target then the THEN section will not be skipped.
- 10 in the IF statement THEN section will not be skipped and the Shoot function will be called, while changing the canAttack to false.
- In the Shoot function on line 1 IF statement THEN section will be skipped since the attackType is "Melee".
- On line 6 in the IF statement THEN section will not be skipped because its attackType is equal to "Melee" and the tower will do damage around itself in a circle form.
- On the Attack function on line 1 in the IF statement THEN section will not be skipped because now canAttack is false, so every second the attackTimer integer will grow.
- On line 3 IF statement THEN section will be started when 2.5 seconds have passed for attackTimer to reach attackCooldown to let the tower attack again and reset the attackTimer count.

Using third set of example data:

- In the Attack function on line 1 in the IF statement THEN section will be skipped because as default the canAttack is equal to true.
- On line 7 it will check IF the tower doesn’t have a target AND IF the queue has more targets inside, if true then it will remove the front queue and make it the new target.
- On line 9 IF the tower has a target then the THEN section will not be skipped.
- On line 10 in the IF statement THEN section will not be skipped and the Shoot function will be called, while changing the canAttack to false.
- In the Shoot function on line 1 IF statement THEN section will not be skipped since the attackType is "Ranged", but since there is no projectileType given it will do nothing.
- On line 6 in the IF statement THEN section will be skipped because its attackType is not equal to "Melee".

5.3 Data Storage

Json:

```
1  {
2   "Username": player1,
3    "map":[
4          {
5              "Map_id": 1,
6              "Difficulty": 3
7          },
8          {
9              "Map_id": 2,
10              "Difficulty": 3
11          },
12          {
13              "Map_id": 3,
14              "Difficulty": 3
15          }
16      ],
```

Figure 10. Example Of Map Saved Data in "map" data

Json:

```
1  {
2    "challenge":[
3          {
4              "challenge_id": 1,
5              "Difficulty": 3
6          },
7          {
8              "challenge_id": 2,
9              "Difficulty": 3
10          },
11          {
12              "challenge_id": 3,
13              "Difficulty": 3
14          }
15      ]
16  }
```

Figure 11. Example Of Beaten Challanges Saved Data in "challenge" data

6 Testing

6.1 Unit Tests

UT001

Title: Download data about what maps player passed

Description: Tests if SELECT query is executed and output data is correct.

Precondition: Information is available on the database.

Assumption: Database is running.

Test Data: username: “test”.

Test Steps:

1. Go to src/backend/webserver/
2. Execute "dotnet test"

Expected Result: Test returns success.

UT002

Title: Post data about map player passed

Description: Tests if INSERT INTO query is executed and input data is correct.

Precondition: Database is running.

Assumption: none

Test Data: username: “test” mapID: 01, difficulty: 1.

Test Steps:

1. Go to src/backend/webserver/
2. Execute "dotnet test"

Expected Result: Test returns success.

UT003

Title: Register new player

Description: Tests if UPDATE query is executed on media URL update attempt.

Precondition: Player already passed this map.

Assumption: none.

Test Data: username: “test” password: “test”

Test Steps:

1. Go to src/backend/webserver/
2. Execute "dotnet test"

Expected Result: Test returns success.

6.2 Integration Tests

IT001

Title: Integration test of login functionality between web server and client side when provided with correct login data.

Description: A registered player should be able to successfully login

Precondition: The player must already be registered.

Assumption: A player’s device is connected to the internet.

Test Steps:

1. Open the game.
2. Open profile menu.
3. Click button "login".
4. In the "Username" field, enter the username of the registered player.
5. In the "Password" field, enter the password of the corresponding player.
6. Click "Ok".

Expected Result: Expected Result: In the account menu, instead of the login and registration window, information about the player’s account should appear

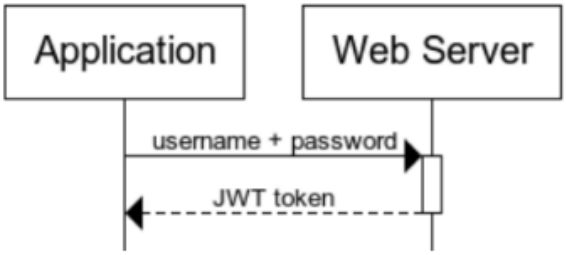

Figure 12. Successful Login Diagram

6.3 System Tests

SYS001

Title: Test for registration of new users.

Description: Player should be able to enter new account data and create an account in the client.

Precondition: Username used for registration should not be already taken and should be entered correctly.

Assumption: Web server, Database, and Client are running

Test Steps:

1. Run application
2. Open account menu by clicking button “Account“ at the main menu
3. Click button “Register”
4. Enter username and password of the new account that is to be created.
5. Click button “Ok” Expected Result: In the account menu, instead of the login and registration window, information about the player’s account should appear

6.4 GUI Tests

Figure 13. Main menu screen

Main menu screen

UIL01 - Main menu screen(Figure 13)
- – Check the button position .
- – Check button image.
- – Check text and font.
- – Check the readability.
- – Test redirection to the Game screen and loading game.

UIL02 - Main menu screen(Figure 13)
- – Check the button position .
- – Check button image.
- – Check text and font.
- – Check the readability.
- – Test redirection to the Game screen.

UIL03 - Main menu screen(Figure 13)
- – Check the button position .
- – Check button image.
- – Check text and font.
- – Check the readability.
- – Test redirection to the Sandbox screen.

UIL04 - Main menu screen(Figure 13)
- – Check the button position .
- – Check button image.
- – Check text and font.
- – Check the readability.
- – Test redirection to the Enless screen.

UIL05 - Main menu screen(Figure 13)
- – Check the button position .

– Check button image.
– Check text and font.
– Check the readability.
– Test if the program is closed.

UIL06 - Main menu screen(Figure 13)
– Check the button position .
– Check button image.
– Test redirection to the Login/Register screen.

UIL07 - Main menu screen(Figure 13)
– Check the button position .
– Check the checkbox.
– Check text and font.
– Check the readability.
– Test if the program is fullscren.

Figure 14. Login/Register screen

Login/Register screen

UIL01 - Login/Register screen(Figure 14)
– Check the text position .
– Check text and font.
– Check the readability.

UIL02 - Login/Register screen(Figure 14)
– Check the button position .
– Check button image.
– Check text and font.
– Check the readability.
– Test redirection to the Login screen.

UIL03 - Login/Register screen(Figure 14)
– Check the button position .

– Check button image.
– Check text and font.
– Check the readability.
– Test redirection to the Register screen.

UIL04 - Login/Register screen(Figure 14)
– Check the button position .
– Check button image.
– Check text and font.
– Check the readability.
– Test redirection to the Main Menu screen.

6.5 Acceptance Tests

AT001

Title: Acceptance test of game for retrieving maps list.

Description: The player should be presented with a list of available maps.

Precondition: The player must have completed at least one level.

Assumption: none

Test Steps:

1. Get to the ‘main menu’ screen.
2. Click the ’play’ button in the main menu screen.

Expected Result: A screen showing the list of maps should be shown.

AT002

Title: Acceptance test of Tower Defense game for retrieving towers list.

Description: The player should be presented with a list of available towers.

Precondition: player should have money for placing the tower

Assumption: none

Test Steps:

1. Get to the ‘main menu’ screen.
2. Click the ’play’ button in the main menu screen.
3. Open any game level.

Expected Result: On the right side of the screen there should be a list of towers that the player can place.

7 Competitive Analysis

The following section analyzes advantageous, and disadvantageous features and the general player feeling of similar "Tower Defense" games with the idea of determining the competitive ability of the game in development.

7.1 Analysis

7.1.1 "Bloons TD 6" by Ninja Kiwi

Advantages:

- Has an enemy targeting system that allows the player to use strategy to beat a level.
- Tower have upgrades that make the gameplay more interesting.

Disadvantages:

- New players get overwhelmed and could stop playing after starting.
- It takes too much time to level up and unlock tower upgrades, forcing the player to play for long time before being able to enjoy the game.

Conclusions and Recommendations

To conclude this report, it needs to be said that it's very important to have a final design in mind and to communicate with the team about the projects future development so that new ideas don't pile on top, hindering the production of said project. That could be done by making a preliminary design and talking about all the features with great detail. While also deviding the team to work on different parts of the project to increase efficiency, but also starting at the basic parts of the project, which will be needed to construct the ideas from ground up rather then starting at something difficult that would make it harder to integrate the basic parts to the project.
Being able to finish most of the funtionalities satisfies the functional requirement priorities, there are still parts that need to be improved and functionalities that still need to be made to contest the more popular games of the same gienre. Functionalities like smarter targeting to help with the towers letting through enemies, tower upgrades for more intresting gameplay for the user and many quality of life content to help the new player understand the game, with tutorials or tooltips. All of these ideas could improve the user enjoyment from the application.

References

[1] *DBPL '17: Proceedings of The 16th International Symposium on Database Programming Languages*, New York, NY, USA, 2017. Association for Computing Machinery.

[2] Daniel Hind and Carlo Harvey. A neat approach to wave generation in tower defense games. In *2022 International Conference on Interactive Media, Smart Systems and Emerging Technologies (IMET)*, pages 1--8, 2022.

[3] Wooseong Jeong and Unil Yun. Development of 2d side-scrolling running game using the unity 3d game engine. In James J. (Jong Hyuk) Park, Yi Pan, Gangman Yi, and Vincenzo Loia, editors, *Advances in Computer Science and Ubiquitous Computing*, pages 132--136, Singapore, 2017. Springer Singapore.

[4] Vid Kraner, Iztok Fister, and Lucija Brezočnik. Procedural content generation of custom tower defense game using genetic algorithms. In Isak Karabegović, editor, *New Technologies, Development and Application IV*, pages 493--503, Cham, 2021. Springer International Publishing.

[5] Simon Liu, Li Chaoran, Li Yue, Ma Heng, Hou Xiao, Shen Yiming, Wang Licong, Chen Ze, Guo Xianghao, Lu Hengtong, Du Yu, and Tang Qinting. Automatic generation of tower defense levels using pcg. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, FDG '19, New York, NY, USA, 2019. Association for Computing Machinery.

[6] D. Polančec and I. Mekterović. Developing moba games using the unity game engine. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1510--1515, 2017.

[7] Qin Wu, Fanglve Zhang, Sihan Zhou, Yuehao Qin, and Xi Wu. Table war: A tower-defense game device with augmented reality projection. In *Proceedings of the Sixth International Symposium of Chinese CHI*, ChineseCHI '18, page 136–139, New York, NY, USA, 2018. Association for Computing Machinery.

Appendices

Example Data

Set 1 of Example Data

attackCooldown: 1.25
attackType: Ranged
projectileType: FireBolt

Set 2 of Example Data

attackCooldown: 2.5
attackType: Melee
projectileType: Null

Set 3 of Example Data

attackCooldown: 1.5
attackType: Ranged
projectileType: Null