Stefan Dimitriu

Mobile Game – Turn Based Strategy

Bachelor Degree in Software Development

January 2019

1. Introduction

1.1 Problem Background

The concept of mobile communication devices came into the common consciousness at the beginning of the 20th century when countries such as The USA and Germany started experimenting with this idea.

The Second World War came with the necessity of better communication between the ground forces and the command unit which caused all major powers to invest in mobile radio technology such as Walkie-Talkies.

Fast forward to 1973, Motorola produced the first ever handheld mobile phone, which at the time, according to Wikipedia, „weighed 1.1 kilograms (2.4 lb.) and measured 23 by 13 by 4.5 centimeters (9.1 by 5.1 by 1.8 in). The prototype offered a talk time of just 30 minutes and took 10 hours to re-charge."

In 1979 the first automatic analogue cellular systems were deployed, this would be later called 1G, followed by 2G(Digital cellular) in 1990, 3G(Mobile broadband) in 2001 and finally 4G in 2009.

Nowadays most people have a mobile phone, according to an article by the Telegraph [8] around 4.9 billion people account for 7.2 billion mobile subscriptions, of which 2.6 billion are smartphone subscriptions. Estimates say that the latter type of subscription will increase to 6.1 billion by 2020 which in my opinion is an incredible number that presents an opportunity for any developer with skills necessary to create an application for a mobile device.

1.2 Problem Statement

What should a mobile game that could take advantage of this growing smartphone market feature?

- What genres should it focus on?
- How do we make players interact with our game on a daily basis?
- How do we keep the data we would receive from players safe and secure?

1.3 Objectives

The purpose of this project is to create a Final Fantasy like Turn Based Game that comes with a login system and a high score leaderboard that will help keep our players active for a long time, while also providing a safe and secure way to become a better player without having to worry about cheaters or hackers.

1.4 Research Questions

To understand what the sub objectives of this project are, a series of research questions will be presented below.

The proposed questions are:

I. What goes into the planning and design of such a project?
II. What are the technologies that should be used for development?
III. Main Features, how were they developed and implemented?
IV. What was the approach in regards to Software Security?
V. How was software testing used to help and improve the development process?

1.5 Relevance

Relevancy, from an academic point of view, comes from the application of knowledge and skills acquired during the 4 years of study at KEA on the subject matter.

From a business point of view this project is relevant due to the high amount of smartphone users that require entertainment in the form of games.

1.6 Feasibility

Description: Produce a smartphone game

Market feasibility:

- 2.9 billion smartphone users
- will reach 6.1 billion in 2020
- high number of competitors that provide low quality games
- Single digit competitors that provide console like quality smartphone games.

Technical feasibility:

- game will be created on the developers home PC
- using coding tools and programs provided by KEA and the developer
- additional laptop for presentation will be acquired at a later date

Financial feasibility:

- a starting capital of 6000DKK will be required to purchase a new laptop for testing and presentations
- Return investment depends on the quality of the end product (0 – 100.000DKK).

Organizational Feasibility:

- One developer
- Software Development focused
- Has created basic games before for desktop systems
- Has worked with phone applications.

1.7 Planning and Designing

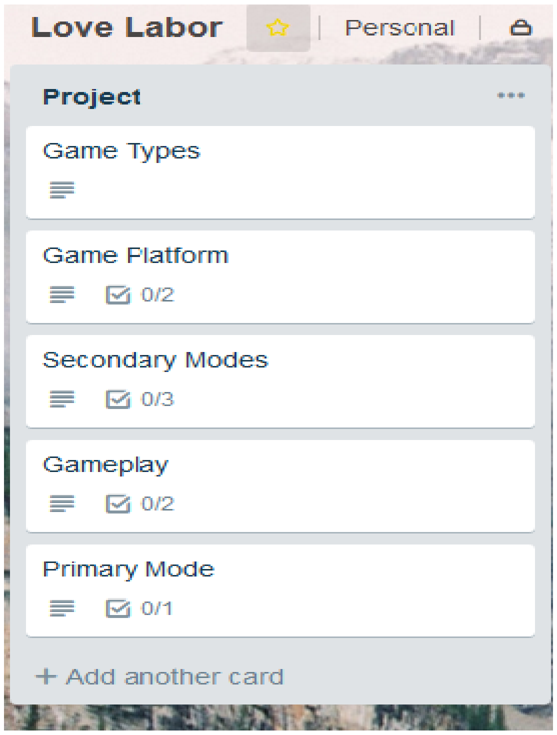
To create a better understanding of what the project as a whole would look like a plan needed to be written down in detail.

To help with this situation Trello was used to separate tasks into different sections that would give a better perspective on the scope of the work involved.

The Trello board was separated into 5 sections:

- Project
- Report
- Working On
- Done
- Curriculum Subjects

The Project section was used to house the planned features of the project.



Love Labor | Personal

Project

Game Types

Game Platform 0/2

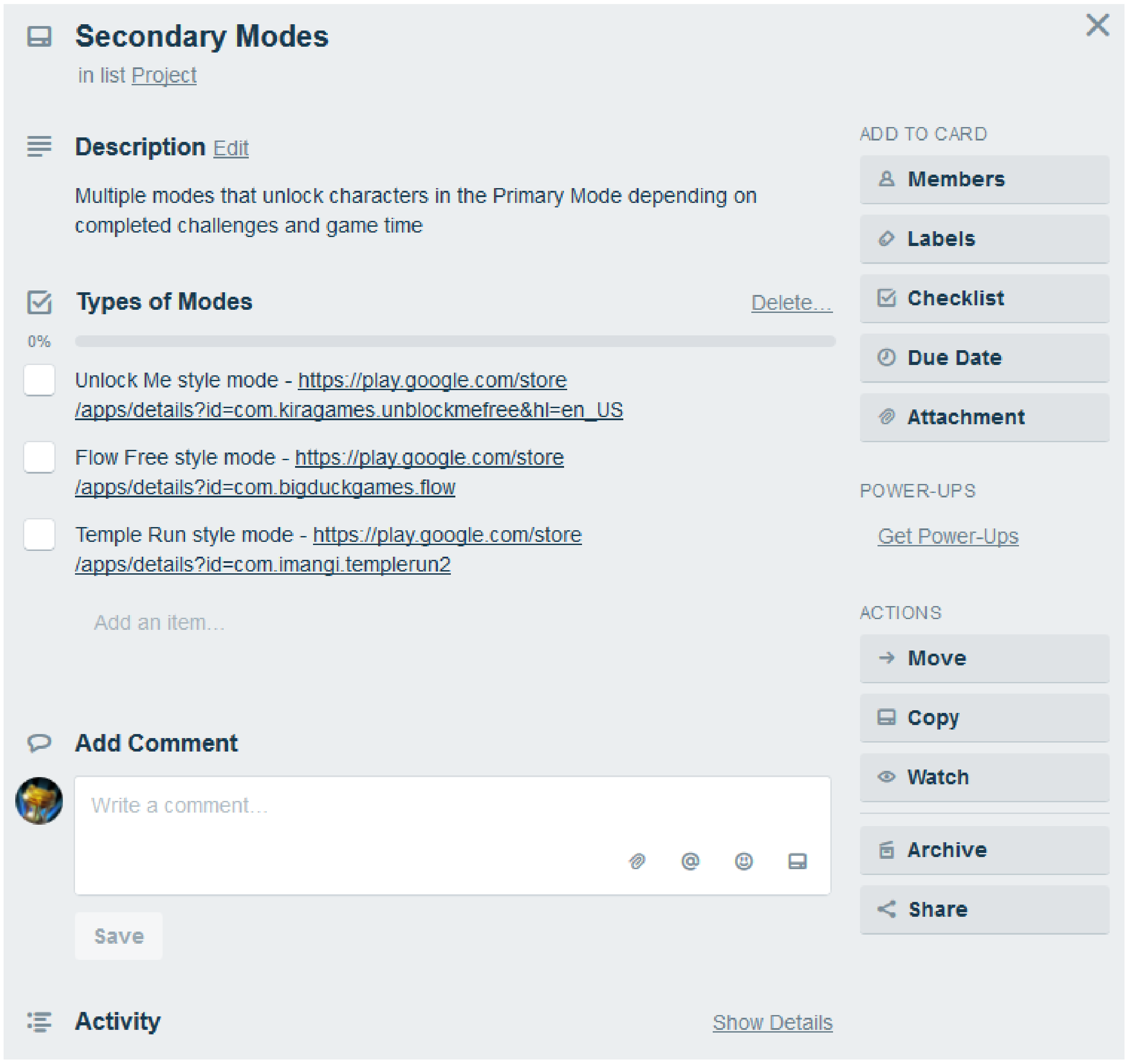
Secondary Modes 0/3

Gameplay 0/2

Primary Mode 0/1

+ Add another card

Each of the feature cards had detailed explanations about what the feature entailed and what the sub tasks were supposed to be.


Secondary Modes

in list Project

Description Edit

Multiple modes that unlock characters in the Primary Mode depending on completed challenges and game time

Types of Modes Delete...

0%

- [ ] Unlock Me style mode - https://play.google.com/store/apps/details?id=com.kiragames.unblockmefree&hl=en_US
- [ ] Flow Free style mode - https://play.google.com/store/apps/details?id=com.bigduckgames.flow
- [ ] Temple Run style mode - https://play.google.com/store/apps/details?id=com.imangi.templerun2

Add an item...

Add Comment

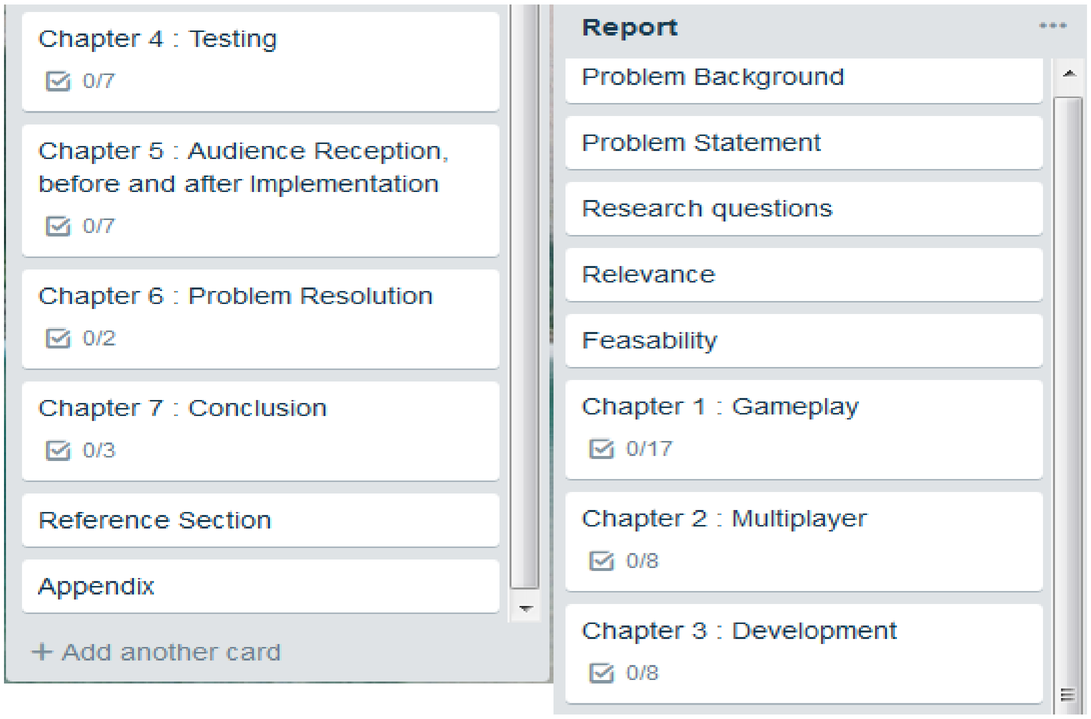
Write a comment...

Save

Activity Show Details

Due to problems during the development process some of the features present on the Trello cards might be different than what is present in the final project.

The Report section was used to detail what each chapter of the report should be about.



Chapter 4 : Testing
0/7

Chapter 5 : Audience Reception, before and after Implementation
0/7

Chapter 6 : Problem Resolution
0/2

Chapter 7 : Conclusion
0/3

Reference Section

Appendix

+ Add another card

Report

Problem Background

Problem Statement

Research questions

Relevance

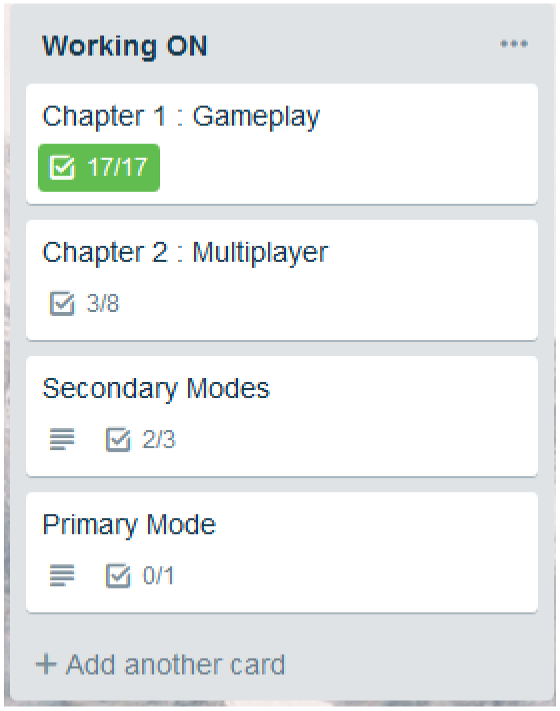
Feasability

Chapter 1 : Gameplay
0/17

Chapter 2 : Multiplayer
0/8

Chapter 3 : Development
0/8

The Working on section was used to show what the tasks currently worked on were.



Working ON

Chapter 1 : Gameplay
17/17

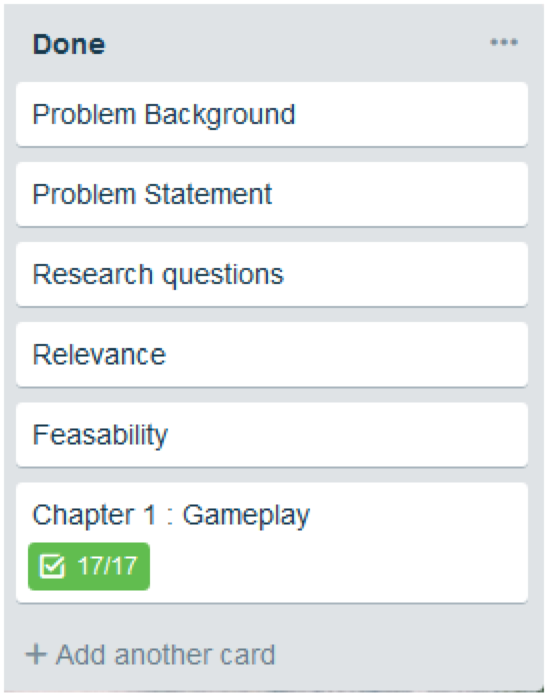
Chapter 2 : Multiplayer
3/8

Secondary Modes
2/3

Primary Mode
0/1

+ Add another card

The Done section was used to house the finished tasks.


Done

Problem Background

Problem Statement

Research questions

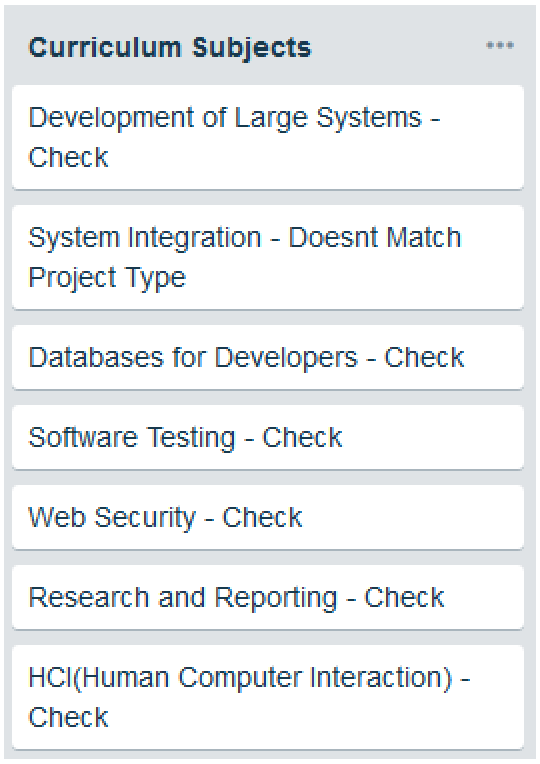
Relevance

Feasability

Chapter 1 : Gameplay
17/17

Add another card

The Curriculum Subjects section existed to remind me of the necessary subjects that the project should cover.


Curriculum Subjects

Development of Large Systems - Check

System Integration - Doesnt Match Project Type

Databases for Developers - Check

Software Testing - Check

Web Security - Check

Research and Reporting - Check

HCI(Human Computer Interaction) - Check

For storage purposes Google Drive was used to house important pdfs and files for this project.

The last tool worth mentioning was Draw.io (https://www.draw.io/) which was used in the making of software diagrams.

1.8 UX Testing and responses to this

When it came to UX testing I decided to prepare a few Task-Based tests in order to find out if:

- The gameplay cycle was easily understood by potential players.
- The UI was user friendly
- The information necessary to create an account was not intruding on people's privacy

In the following paragraphs we will analyze the data from the 3 result sets presented above.

The gameplay cycle involves playing a match, gaining experience depending on the results of that match, raising the stats of each individual character, and unlocking new characters based on the high scores obtained by the player. The subjects were asked to do exactly this.

For this test 3 different people were asked to participate:

- Subject 1(S1) that plays this specific genre of game(RTS – Real Time Strategy, TBS – Turn Based Strategy, MOBA - Multiplayer online battle arena)
- Subject 2(S2) that plays entirely different genres of games(FPS – First Person Shooter, MOBA - Multiplayer online battle arena)
- Subject3(S3) that does not play games

S1 was immediately familiar with the games concept and gameplay cycle, though they were surprised to see that your overall high score was used to determine what characters you could unlock and use in matches. They also mentioned that the gameplay while easy and to the point felt too simple and wished that there was more to the combat system.

Later during the development process this raised concerns about the games security. Theoretically, if you somehow modify your own personal high score by gaining access to the database you could unlock everything from the start and skip the entire gameplay loop.

S2 brought some interesting remarks and opinions regarding the gameplay loop. When asked about the gameplay they said they enjoyed how simple and to the point the matches were and thought that adding more complex features would turn away players like them who prefer other genres of games.

Due to S2s opinion, the gameplay loop was left untouched, this way, players from all genres can enjoy this game in small, 5 min bursts while on lunch break and so on.

S3 had never played a mobile/desktop/console game before. Their experience with the game would determine if the gameplay loop was accessible for everyone.

After a few matches and some explanations about certain mechanics, S3 was able play without any assistance. While S3s experience was positive, this proved that the game required a tutorial to explain the mechanics and help with improving the usability of the project.

The second Usability Test consisted of asking subjects to use the UI given to them to register and login. The UI was made to be similar to most other Login/Register UIs.

S1 and S2 were familiar with the process of login and registering for an account as that is something common these days.

S3 required an explanation of what an account was and what is was needed for.

All of the subjects had no problems with the amount of information requested by the registration process.

2.Technologies, why and how they were used

2.1 C#

During my 4 years at KEA I learned how to code in many different languages, such as Java, JavaScript, C#, HTML, CSS and so on.

During my internship project at “Kea Project Lab” which focused on games in VR, I developed a deeper understanding of C#, and also Unity which I will be discussing later in this chapter. When the time for my final project came I knew I wanted to make something that made use of my skills in C#.

Due to its similarity to Java and C++ I had no problems in quickly getting used to the small differences between them.

In the following paragraph a short definition and exposition about C# will be presented.

According to Technopedia - “C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language.” [1]

“C# has a strict Boolean data variable type, such as bool, whereas C++ bool variable types may be returned as integers or pointers to avoid common programming errors.”[1]

“C# automatically manages inaccessible object memory using a garbage collector, which eliminates developer concerns and memory leaks.”[1]

"C# type is safer than C++ and has safe default conversions only (for example, integer widening), which are implemented during compile or runtime."[1]

To use C# to its full potential Visual Studio 2017 was used as an IDE (Integrated development environment) to process the code.

2.2 Unity

When considering what type of engine my game should use I slimmed down my list to 3 possibilities "Game Maker", "Unreal Engine 4" and "The Unity Engine". Due to me having a past of working in Unity before, the first time being when I made a game in The Unity Engine for my final project during my AP Degree and the second time during my recent internship at KEA Project Lab were we had to create a VR Experience, I decided that making another project in Unity would benefit me since I already have previous experience.

The Unity Engine in some ways could be compared to a Lego set. You take different Lego pieces and then you write code that you attach to them, that tell them what to do, how to act in certain situations and so on.

In order to place and use these components The Unity Editor is used.

The Unity Editor gives the programmer access to the Unity Engine and its compiler which can build a project created by you in any of the major Oss (Operating Systems). This projects final form was built for Android and PC.

2.3 .NET

While researching how to attach a leaderboard/login system to this game I read on different forums that connecting the database directly to the games interface would leave the database open to different software attacks.

A suggestion to this problem, which was in line with the objectives of this project, was adding a middle-ware component (an API) that would serve to connect the database and the game through a secure channel.

At the time, due to my previous experience in using .NET (.NET 4.6.2) in school projects and also using C# as my primary coding language I thought it was obvious to pick this framework.

To use properly use .Net, Visual Studio 2017 was used as an IDE (Integrated development environment).

2.4 SQL

Since I wanted to have a login system/leaderboard it made sense to add a database. When it comes to designing and implementing database I have the most experience in SQL.

To access and design the database tables and all the other information Microsoft SQL Server Management Studio 2017 (SSMS17) was used in combination with Visual Studio 2017 and SQL Server 2016/2014.

2.5 HTML

HTML was used for writing the server requests.

3. Analysis of the main project features

The following chapter will analyze and describe the implementation of the features present in the project.

The main features of the project are:

- Registering an account
- Login into an account
- Turn based combat gameplay
- Single player matches
- A Leaderboard that updates depending on how well you do

3.1 Game UI and Models(C# & Unity)

A game from a programming perspective is divided into Visual components and logical components:

- The Visual components being the UI and the in-game Models (3D or 2D).
- The Logical components being the scripts attached to different Models or UI elements

The first thing when it comes to designing the Visual parts of a game is to understand what features should be present from this point of view.

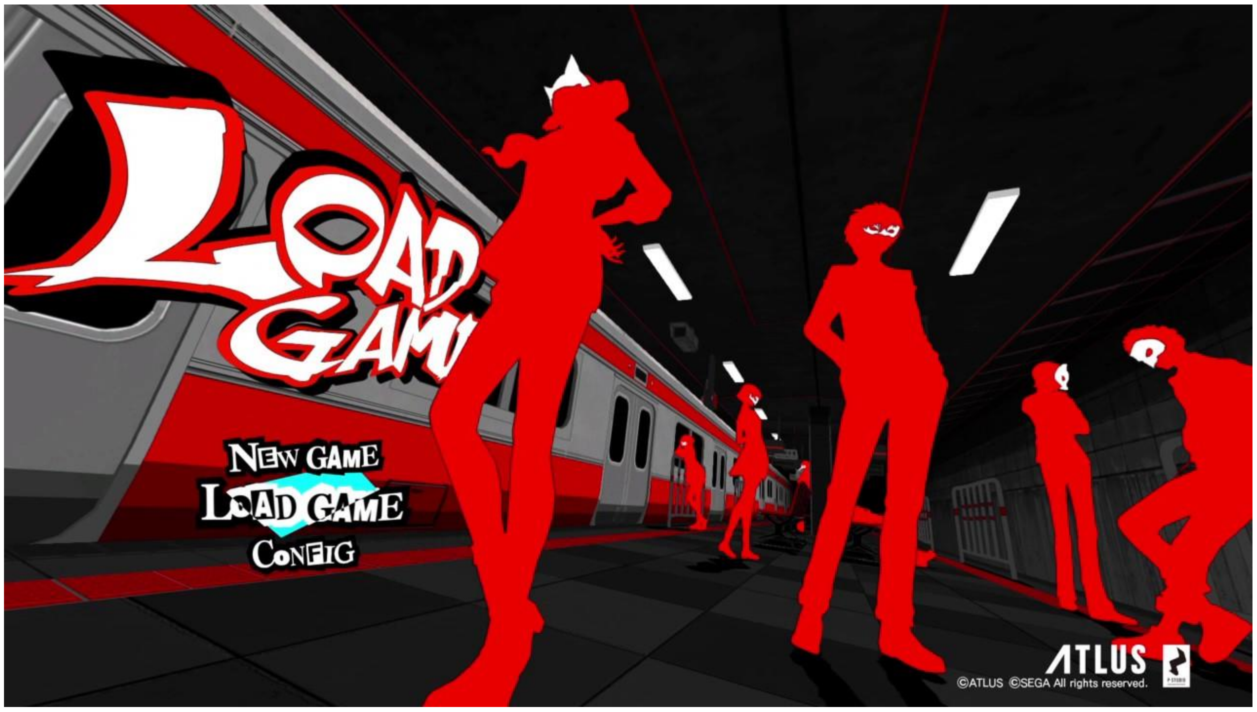
The main ones are:

- A Menu
- Inanimate objects
- Player objects
- Background art

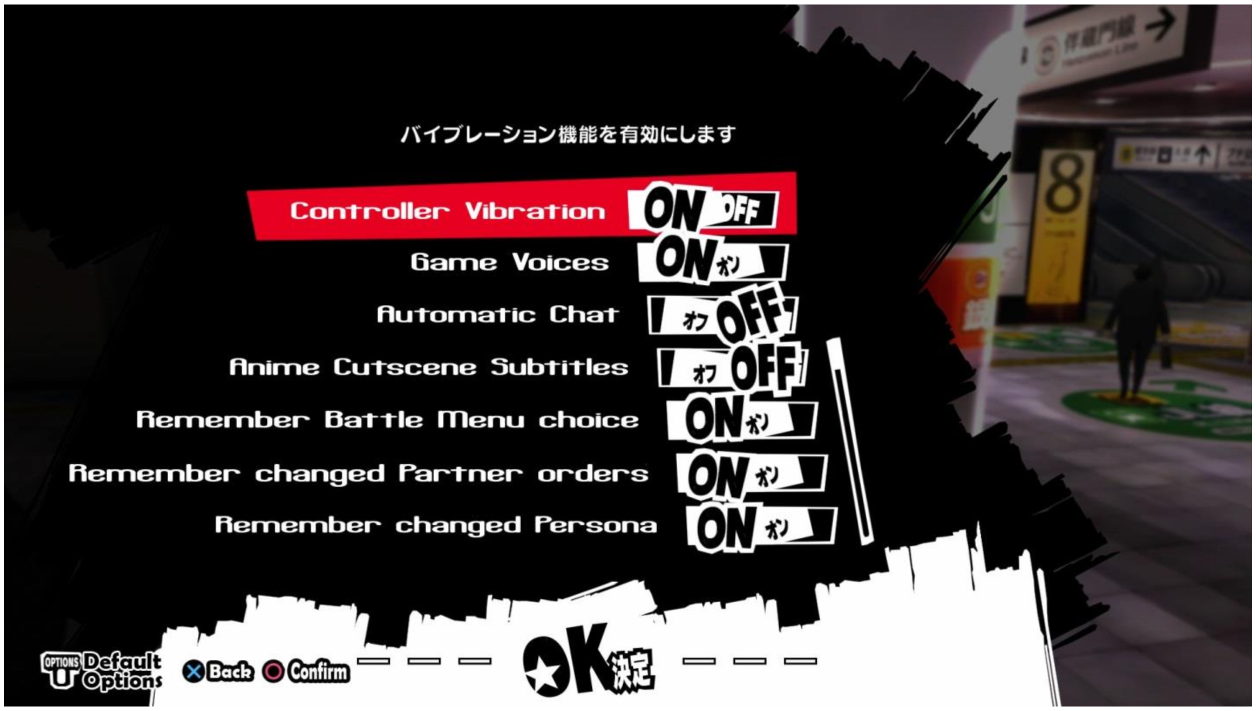
To design a proper menu that is in line with the concepts of user-friendliness and accessibility I looked at the most popular games of the year from both the mobile, console and PC world.

Each of these 3 different sections of the gaming world takes a different approach to the designing of a menu screen.

The console world has a minimalistic take on the menu options. Their purpose is to get the player in the game as soon as possible and leave more detailed options for later in the game.



When a player meets a new mechanic the game usually has an option to change the way the player interacts with it, be it by inverting the way the camera shifts or changing if you should mash the buttons or keep one pressed long enough for an action to occur.



バイブレーション機能を有効にします

Controller Vibration ON OFF

Game Voices ON オン

Automatic Chat オフ OFF

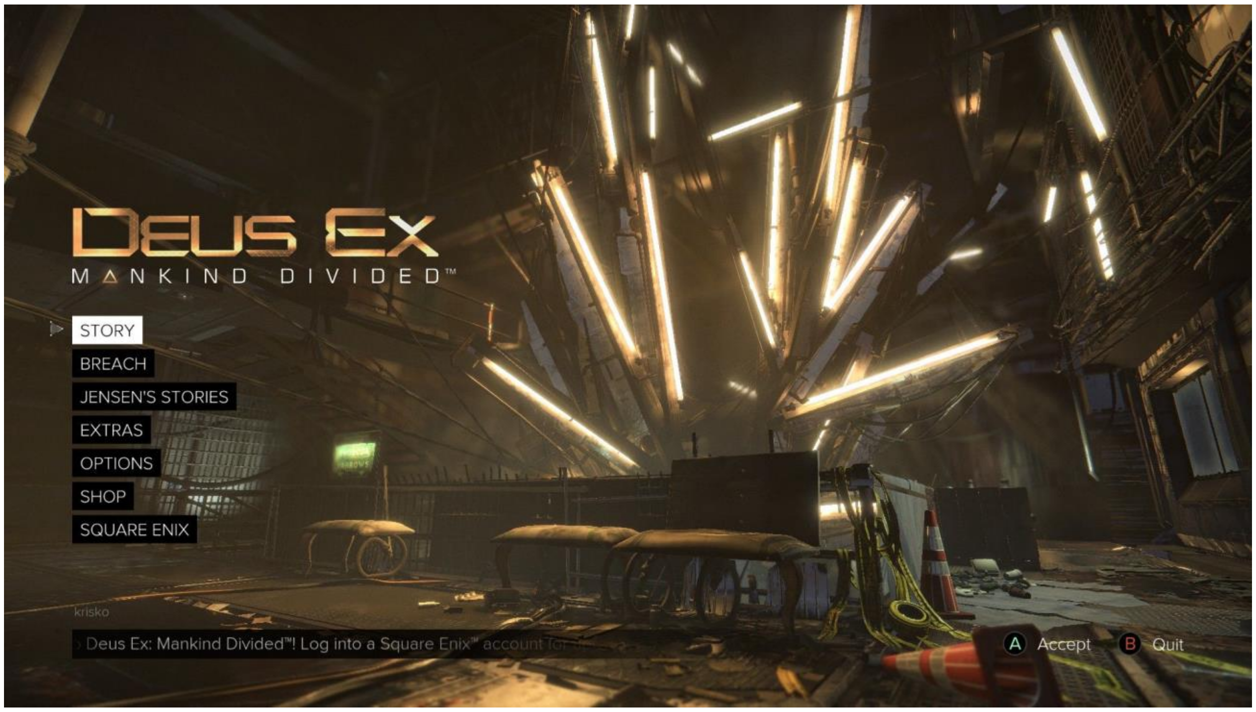
Anime Cutscene Subtitles オフ OFF

Remember Battle Menu choice ON オン

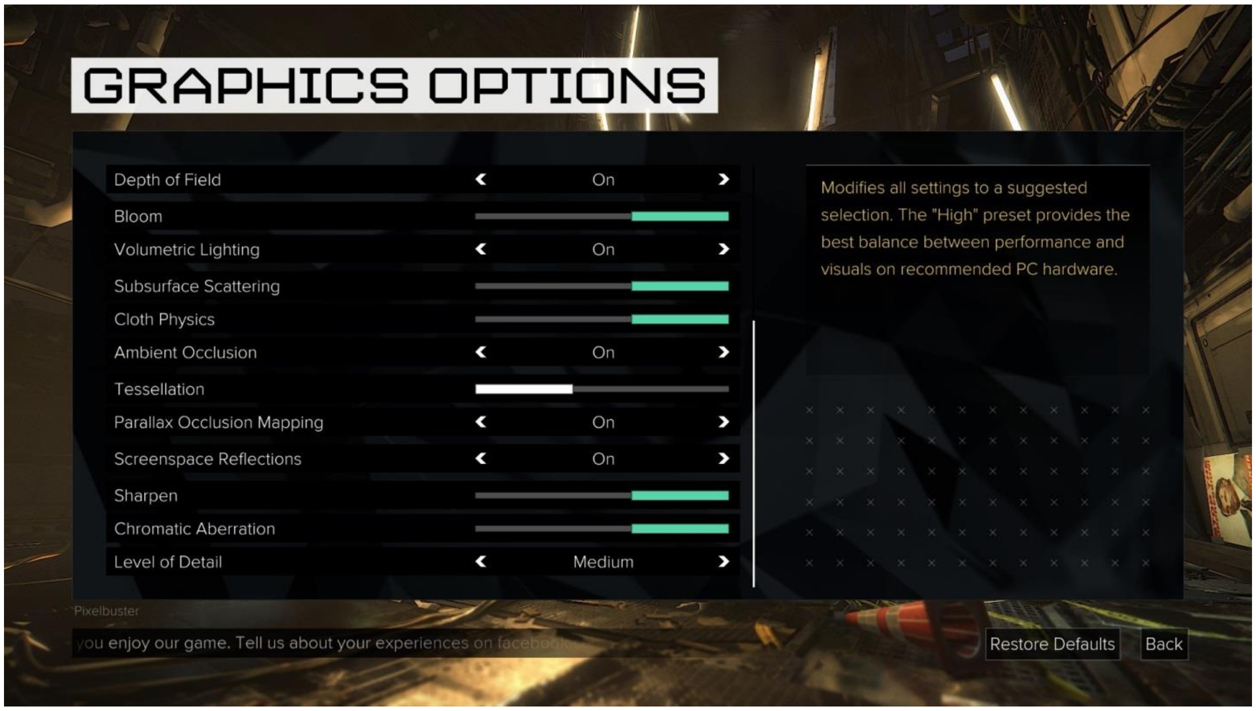
Remember changed Partner orders ON オン

Remember changed Persona ON オン

The PC gaming world prefers to give you all the options and configuration options from the start since most of these games and their performance is dependent on each player's hardware.



The main difference between console games and PC games, when it comes to menus, is that PC offers the option to change the games graphics and as such modify the performance and visual quality of the game.



GRAPHICS OPTIONS

Depth of Field	On
Bloom	
Volumetric Lighting	On
Subsurface Scattering	
Cloth Physics	
Ambient Occlusion	On
Tessellation	
Parallax Occlusion Mapping	On
Screenspace Reflections	On
Sharpen	
Chromatic Aberration	
Level of Detail	Medium

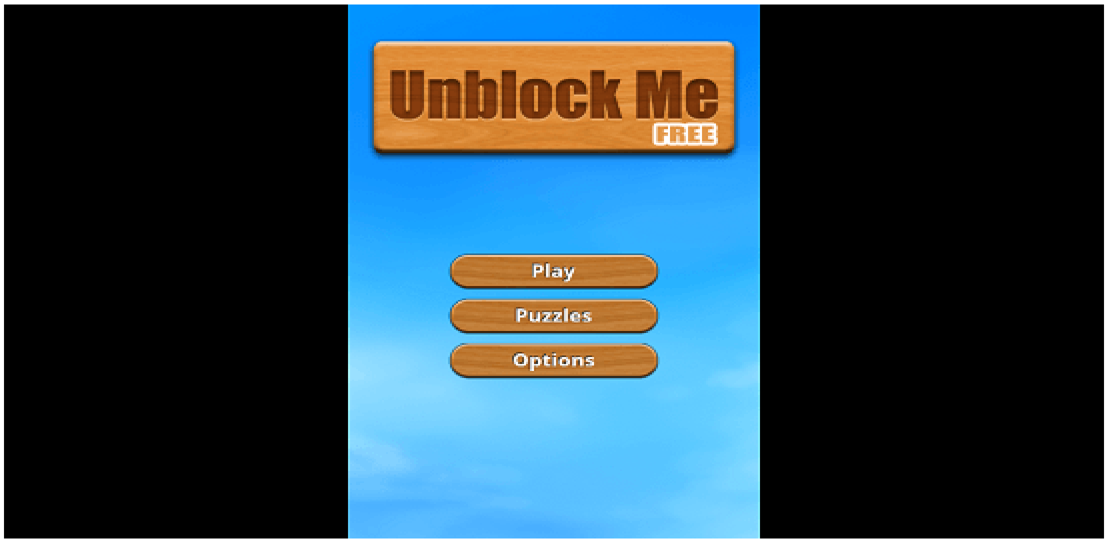
Modifies all settings to a suggested selection. The "High" preset provides the best balance between performance and visuals on recommended PC hardware.

Pixelbuster

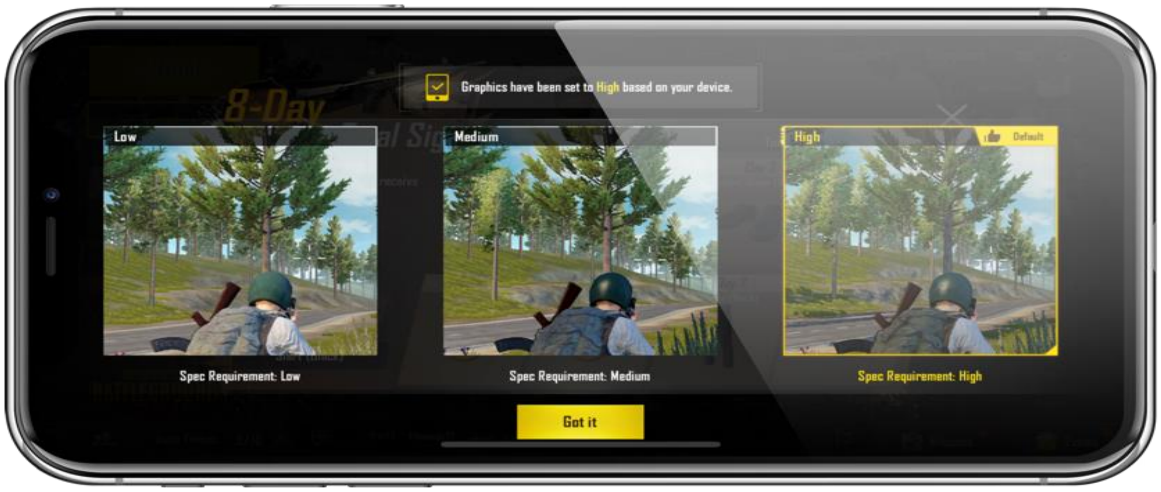
ou enjoy our game. Tell us about your experiences on faceboo

Restore Defaults | Back

The mobile world is similar to the console way of creating menus, this is due to mobile phones most of time not having the power a console or pc would have and as such can`t provide options to modify the in-game graphics or change the performance on the application.



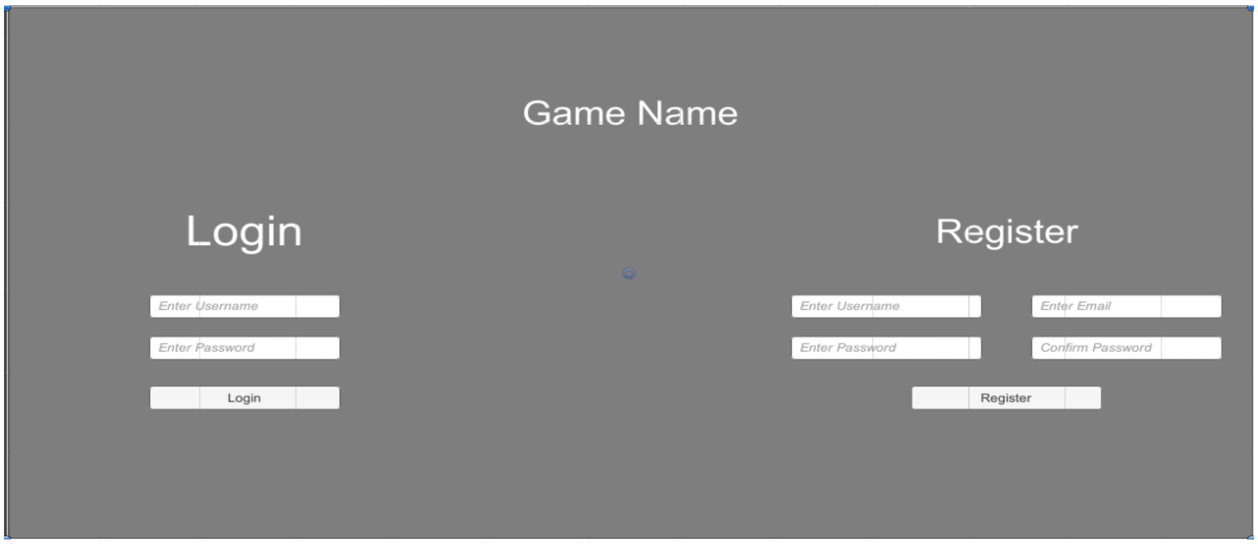
Recently that has changed somewhat due to the use of engines such as the Unity Engine and Unreal Engine 4 which provide phones, with top-end hardware, the ability to change graphics options in favor of either more performance or better graphical fidelity.



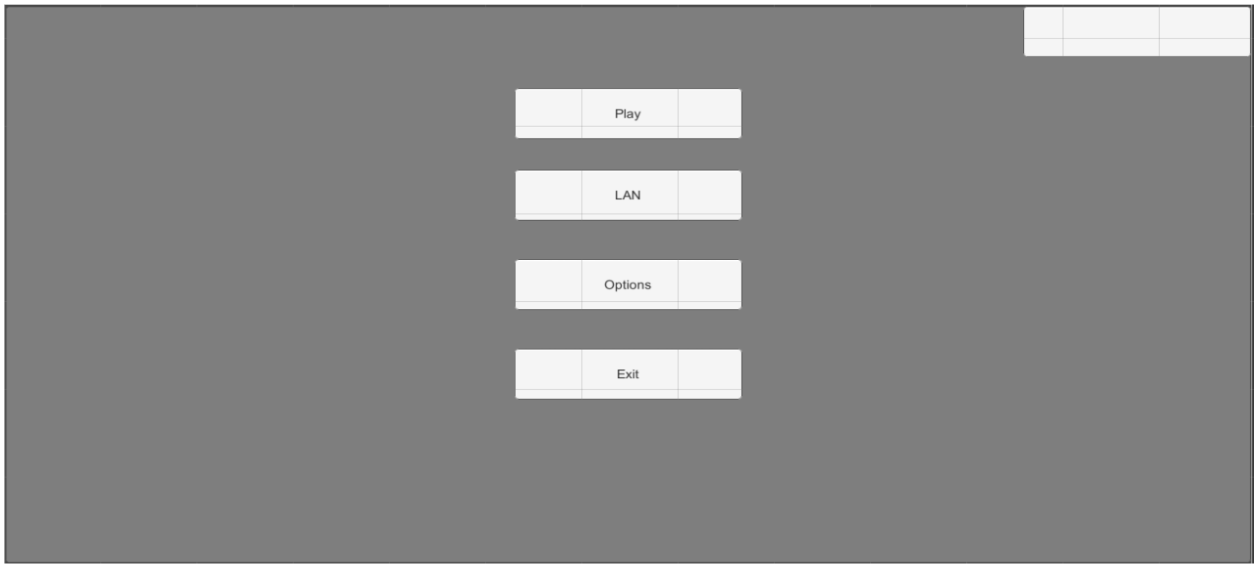
After researching the way a menu should look and behave the decision was made for this games menu to resemble a mobile phone menu.

At the end of the development process the Menu was composed of 10 “Scenes” that contained the options of each gameplay feature.

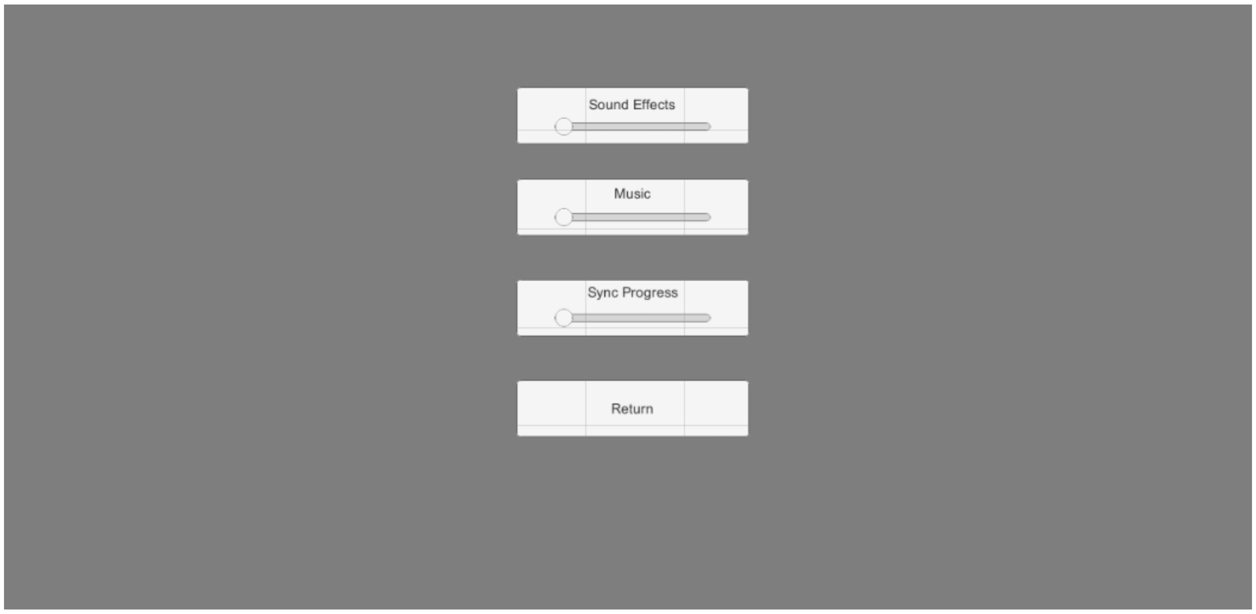
The “Login and Register” menu is used by the player to register or log into an account.



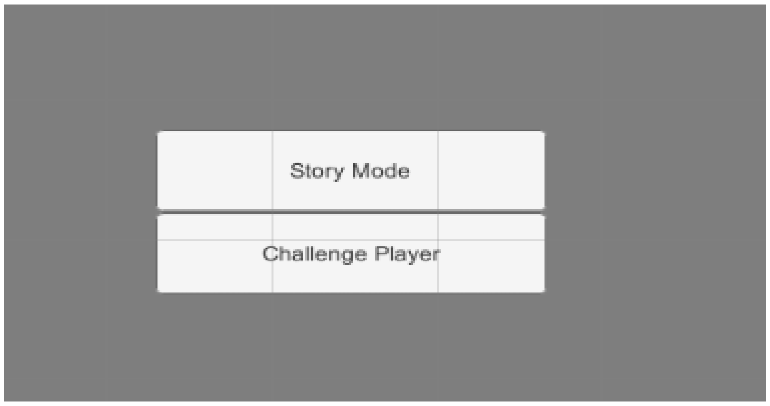
The “Start“ menu allows the player to choose if he wants to play alone or with his friends, access the Options menu or Exit the game.



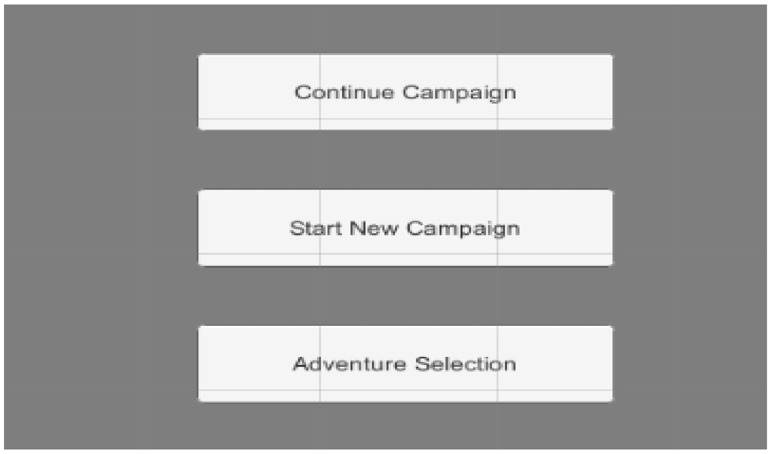
In the “Options” menu the player can change the volume of the music or of the sound effects, sync his player data with the server and save his progress manually or return to the “Start Menu”.



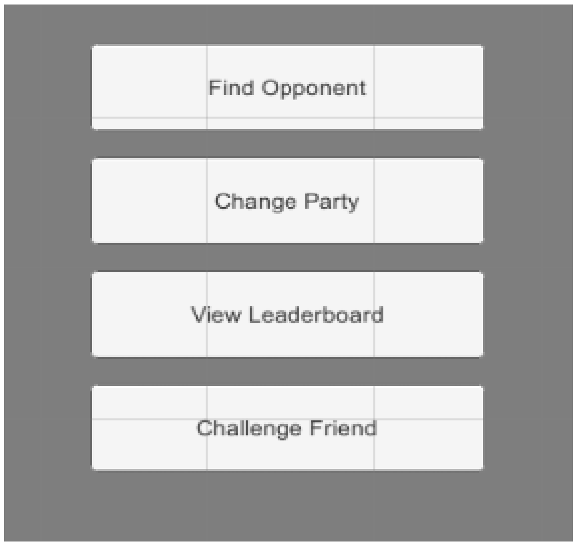
The “Play” menu offers a choice between “Story Mode” and “Challenge Player”.



The “Story Mode” is an on rails tutorial that helps players become familiar with the mechanics of the game and also offers a basic set of unlockable characters.

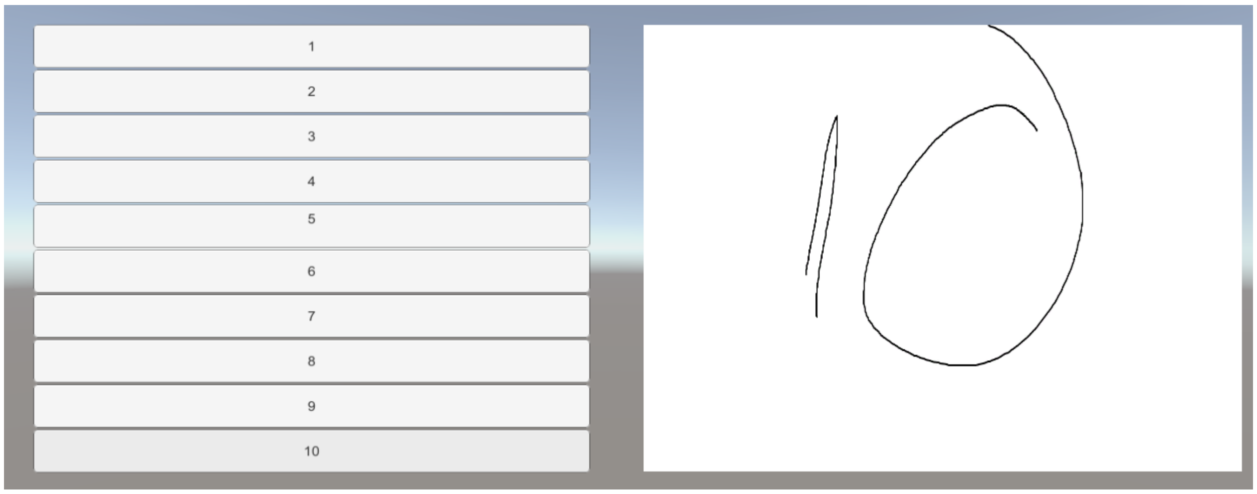


The “Challenge Player” option is used to find an online opponent to battle with your customized party.

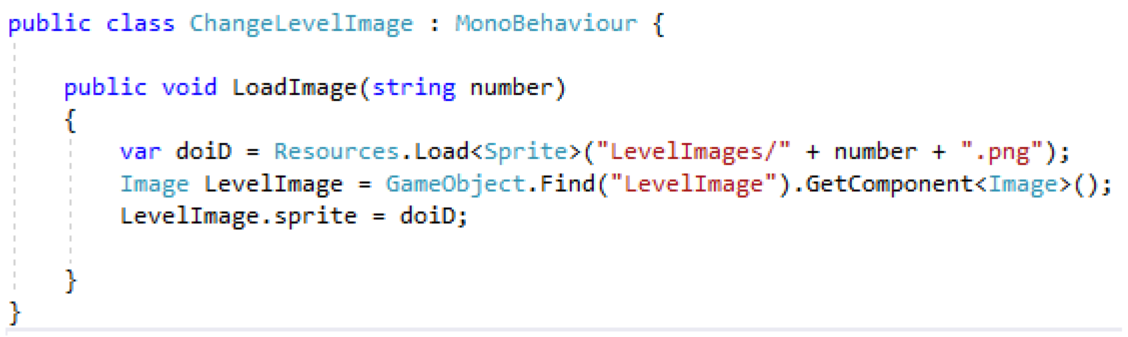


Both the “Story Mode” and “Challenge Player” come with their own set of options which allow the customization of your 3 member party before entering a match. The “Challenge Player” mode also comes with an option to view the Leaderboard and see where you stack against other players.

In the “Level Selection” menu contained in the “Story Mode” menu the player is able to choose what level he wants to replay for a better score which can under certain circumstances unlock new characters.

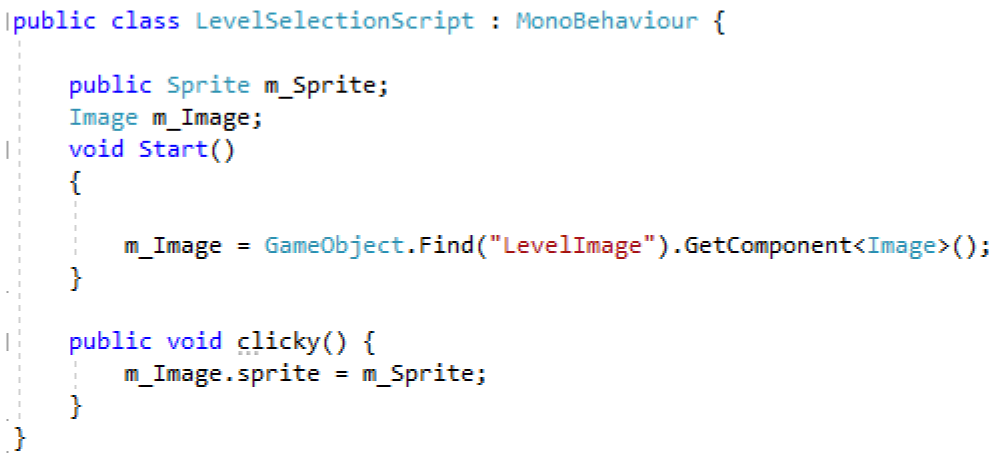


The panel on the right side of screen will contain a preview of the battlefield on which the character will fight.



```
public class ChangeLevelImage : MonoBehaviour {

    public void LoadImage(string number)
    {
        var doiD = Resources.Load<Sprite>("LevelImages/" + number + ".png");
        Image LevelImage = GameObject.Find("LevelImage").GetComponent<Image>();
        LevelImage.sprite = doiD;

    }
}
```

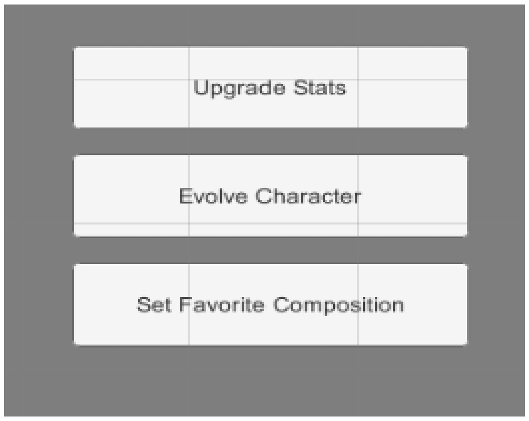
The code shown above is used to load the level images from a local folder.



```
public class LevelSelectionScript : MonoBehaviour {

    public Sprite m_Sprite;
    Image m_Image;
    void Start()
    {

        m_Image = GameObject.Find("LevelImage").GetComponent<Image>();
    }

    public void clicky() {
        m_Image.sprite = m_Sprite;
    }
}
```

When the player clicks on a level button this script executes and changes the level image.

One final menu scene is the “Party” menu in which players can change the values of their characters, change their appearances and set their favorite compositions for online matches, this is done to help players who want to play a quick 5 minute game without having to set up their composition.



3.2 Turn Based Combat

While researching the genres of games that would work on mobile the most popular ones at the time were “Runners”, ”Puzzles” and ”Strategy”.

“Easy to learn, hard to master”, this phrase describes the most successful games out there, be them on pc or mobile. This type of game focuses on giving you easy to understand mechanic that then take a long time to master with the main goal of the player being to become better, improve their score/stats and become the top player in a certain section.

This type of game bets on the innate wish for competition that people have. Being the best in a field is something that many people wish for. When the word strategy is used a lot of people immediately jump on the band-wagon thinking that this experience will help them prove to others or to them that they are better in some way shape or form. This type of game can help people gain confidence and become better not only at the game but in life as well.

Examples of such games are League of Legends, Counter Strike Global Offensive, StarCraft and many others which have an ESports scene.

In order to combine all the concepts presented above the idea of a Turn Based combat system was born, a simple to learn but hard to be good at game.

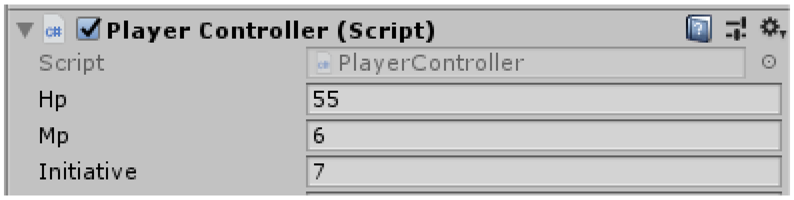
To design and implement such a system a developer would need to think of the following things:

- How do you decide the order in which players act?
- What abilities should the players have access to?
- What levels do players get access to better abilities?
- How to balance these abilities?
- How many characters players have control of at a time?
- Can the players change characters during a match?
- What should the rewards based on the result of the match be?

- Should the damage calculations be made using player devices or should the web server make these calculations?

The game developed for this project pits 2 players against each other. Both combatants use 3 unique characters from their character pool that can't be changed during a match. Each character has levels, abilities and stats that change depending on the player performance during matches.

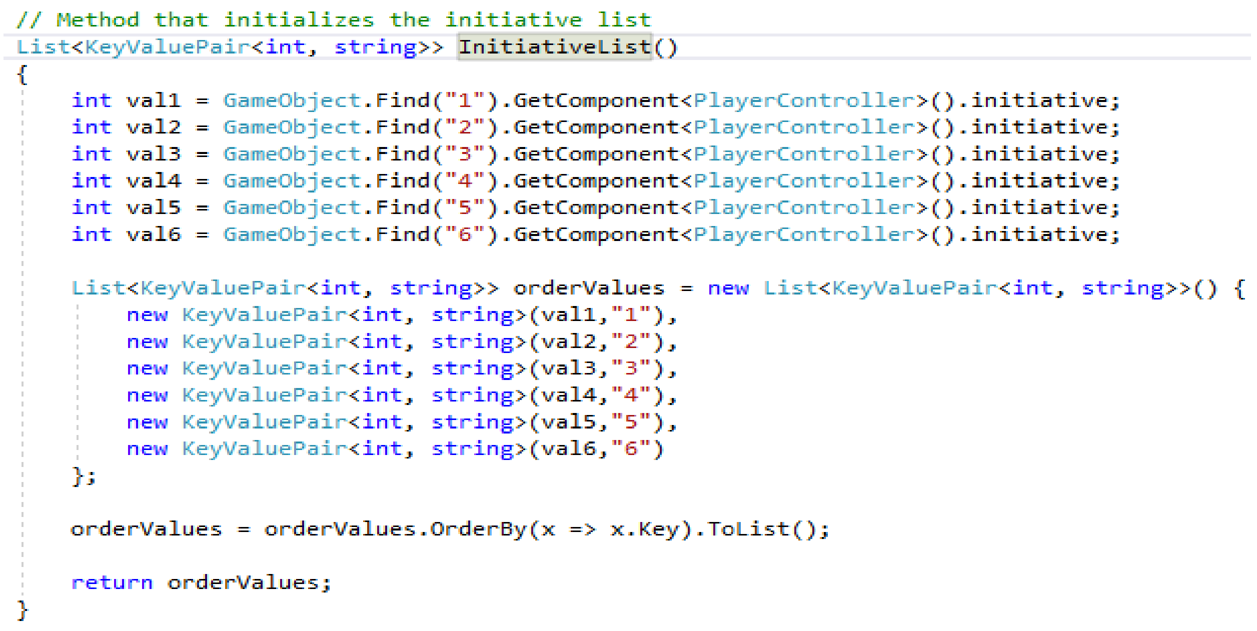
Each character has 3 main values that change during the course of a match; HP (Hit Points), MP (Mana Points) and Initiative.



Script	PlayerController
Hp	55
Mp	6
Initiative	7

In games of this type you need to design a mechanic that decides what the order in which characters get to act is, that mechanic in this case is the value field of initiative.

At the start of match the game takes each characters Initiative value along with the name of the object that the value is attached to and places them into a "KeyValuePair" List which is then sorted in ascending order.



```
// Method that initializes the initiative list
List<KeyValuePair<int, string>> InitiativeList()
{
    int val1 = GameObject.Find("1").GetComponent<PlayerController>().initiative;
    int val2 = GameObject.Find("2").GetComponent<PlayerController>().initiative;
    int val3 = GameObject.Find("3").GetComponent<PlayerController>().initiative;
    int val4 = GameObject.Find("4").GetComponent<PlayerController>().initiative;
    int val5 = GameObject.Find("5").GetComponent<PlayerController>().initiative;
    int val6 = GameObject.Find("6").GetComponent<PlayerController>().initiative;

    List<KeyValuePair<int, string>> orderValues = new List<KeyValuePair<int, string>>() {
        new KeyValuePair<int, string>(val1,"1"),
        new KeyValuePair<int, string>(val2,"2"),
        new KeyValuePair<int, string>(val3,"3"),
        new KeyValuePair<int, string>(val4,"4"),
        new KeyValuePair<int, string>(val5,"5"),
        new KeyValuePair<int, string>(val6,"6")
    };

    orderValues = orderValues.OrderBy(x => x.Key).ToList();

    return orderValues;
}
```

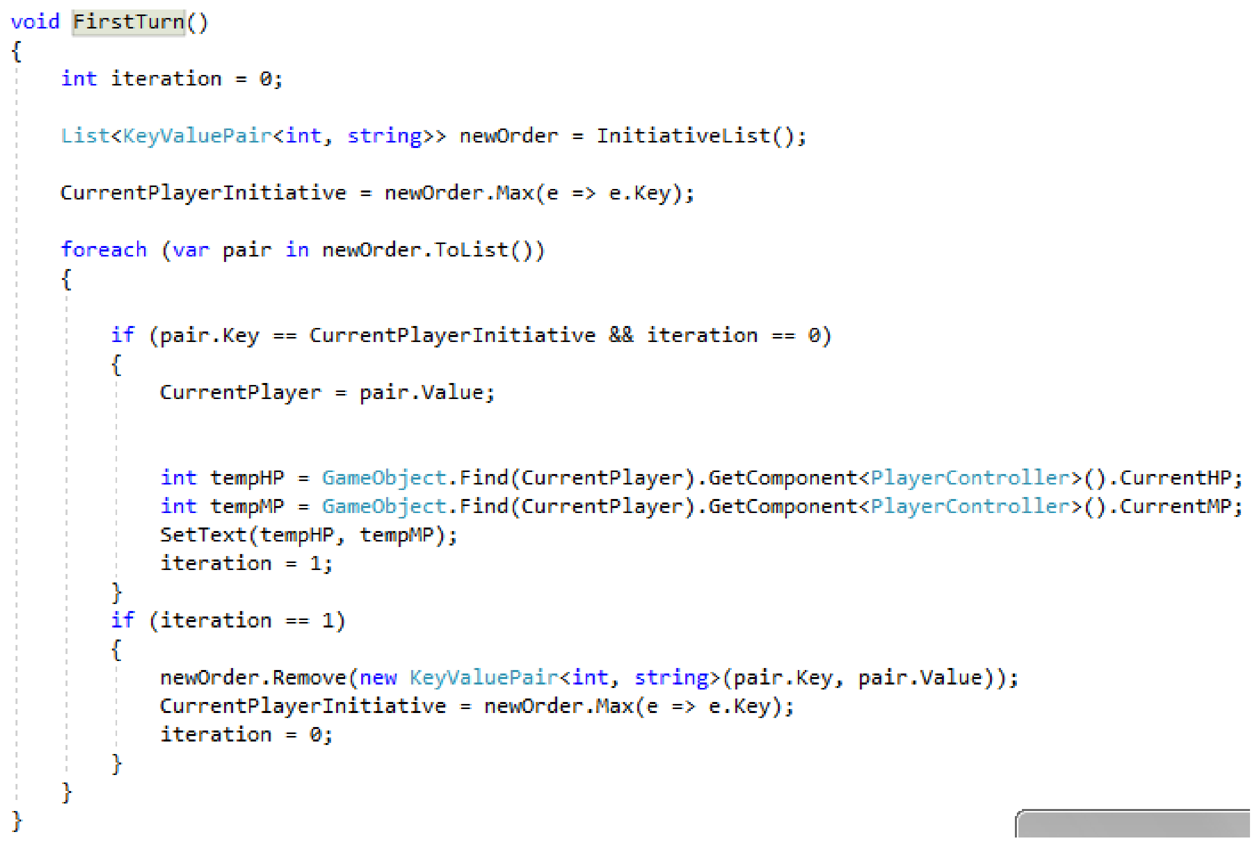
The ordered list is then used in Turn System which will be discussed in the following section.

The way turns work is as follows:

- A new turn begins
- The player with the highest initiative that hasn’t taken a turn yet gets to take an action
- The player takes an action either by attacking and damaging an enemy or restoring HP to an ally
- The player then hits End turn
- The process repeats itself

To make this Turn system work, 2 main methods and 6 secondary methods were used.

The first main method is called **FirstTurn** and is used to determine which character goes first at the beginning of each round.

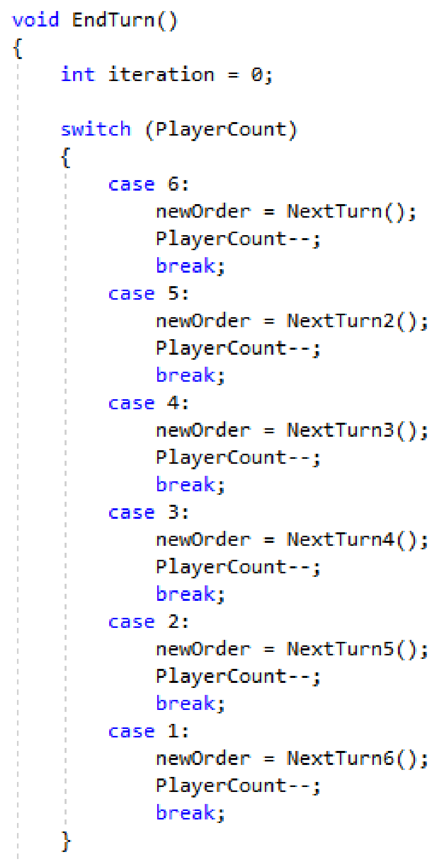


```
void FirstTurn()
{
    int iteration = 0;

    List<KeyValuePair<int, string>> newOrder = InitiativeList();

    CurrentPlayerInitiative = newOrder.Max(e => e.Key);

    foreach (var pair in newOrder.ToList())
    {

        if (pair.Key == CurrentPlayerInitiative && iteration == 0)
        {
            CurrentPlayer = pair.Value;


            int tempHP = GameObject.Find(CurrentPlayer).GetComponent<PlayerController>().CurrentHP;
            int tempMP = GameObject.Find(CurrentPlayer).GetComponent<PlayerController>().CurrentMP;
            SetText(tempHP, tempMP);
            iteration = 1;
        }
        if (iteration == 1)
        {
            newOrder.Remove(new KeyValuePair<int, string>(pair.Key, pair.Value));
            CurrentPlayerInitiative = newOrder.Max(e => e.Key);
            iteration = 0;
        }
    }
}
```

This method involves using the list returned by the **InitiativeList** function and finding the MAX value from the aforementioned list, then using that value to find the name of the object that has that MAX value. This allows us to know which character starts first and also update our "Current Player" panel with that information. At the end of the method we remove this character from the turn order until the next round.

When the player wants to end a characters turn they will press the End Turn button on the right side of the screen and next player in the initiative order will take its turn.

The next character in the initiative order is then decided by the **EndTurn** method.



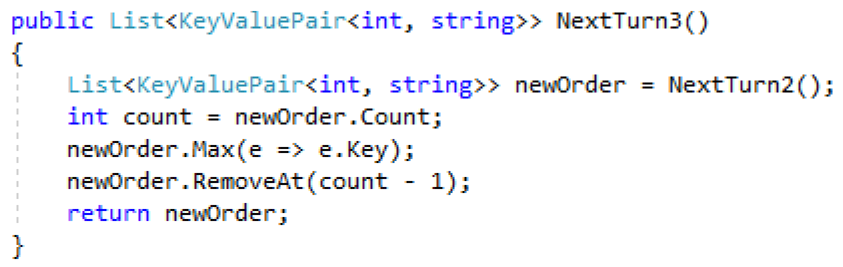
```
void EndTurn()
{
    int iteration = 0;

    switch (PlayerCount)
    {
        case 6:
            newOrder = NextTurn();
            PlayerCount--;
            break;
        case 5:
            newOrder = NextTurn2();
            PlayerCount--;
            break;
        case 4:
            newOrder = NextTurn3();
            PlayerCount--;
            break;
        case 3:
            newOrder = NextTurn4();
            PlayerCount--;
            break;
        case 2:
            newOrder = NextTurn5();
            PlayerCount--;
            break;
        case 1:
            newOrder = NextTurn6();
            PlayerCount--;
            break;
    }
```

This method uses a switch case due the number of characters and the necessity of a recursive way of deciding whose turn it is.

This switch case contains 6 situations, one for each character.

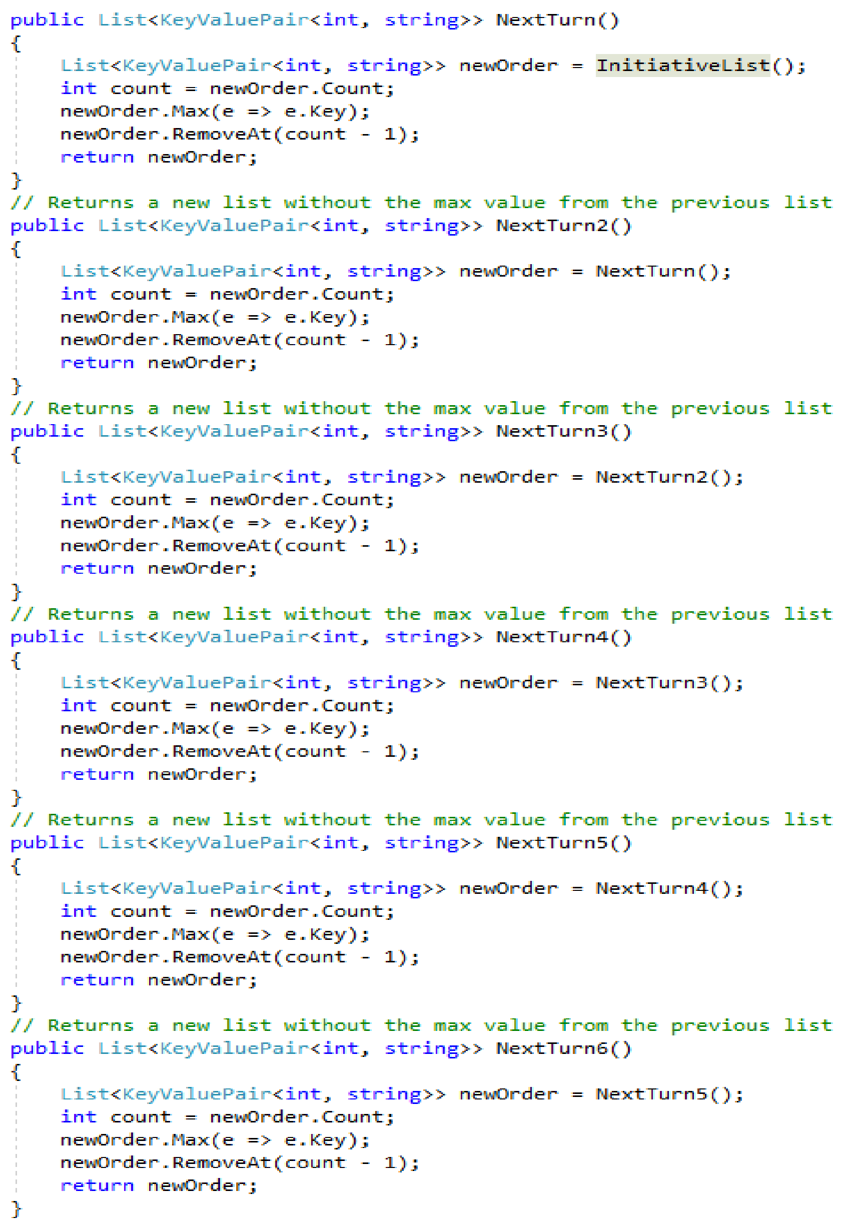
A variable with the name of “PlayerCount” was used to determine what character is taking its turn.

Let’s imagine that currently Character 3 is taking its turn, in that case the “PlayerCount” has a value 3. The switch case then jumps to “case 3” and calls the secondary method NextTurn3.



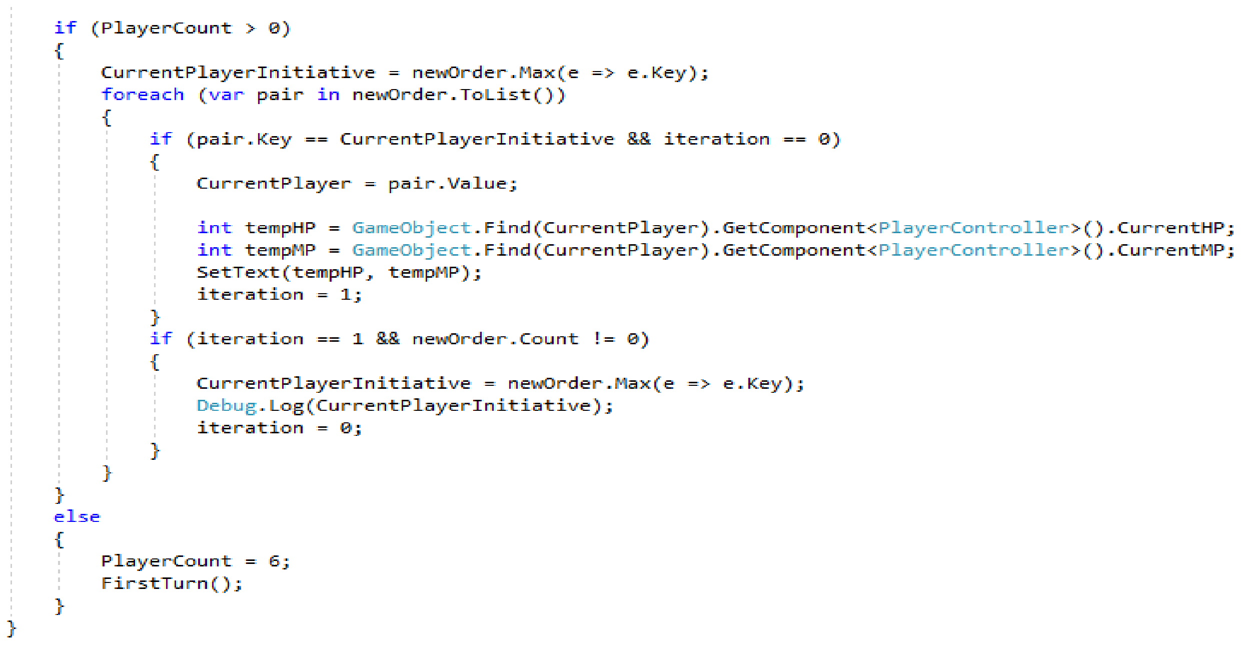
```
public List<KeyValuePair<int, string>> NextTurn3()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn2();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
```

The purpose of this method is to find and eliminate the value that was used to determine the current characters turn and return a new list without that value.



```
public List<KeyValuePair<int, string>> NextTurn()
{
    List<KeyValuePair<int, string>> newOrder = InitiativeList();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
// Returns a new list without the max value from the previous list
public List<KeyValuePair<int, string>> NextTurn2()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
// Returns a new list without the max value from the previous list
public List<KeyValuePair<int, string>> NextTurn3()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn2();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
// Returns a new list without the max value from the previous list
public List<KeyValuePair<int, string>> NextTurn4()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn3();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
// Returns a new list without the max value from the previous list
public List<KeyValuePair<int, string>> NextTurn5()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn4();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
// Returns a new list without the max value from the previous list
public List<KeyValuePair<int, string>> NextTurn6()
{
    List<KeyValuePair<int, string>> newOrder = NextTurn5();
    int count = newOrder.Count;
    newOrder.Max(e => e.Key);
    newOrder.RemoveAt(count - 1);
    return newOrder;
}
```

There are specific methods for each of cases mentioned above as shown in the photo.



```
    if (PlayerCount > 0)
    {
        CurrentPlayerInitiative = newOrder.Max(e => e.Key);
        foreach (var pair in newOrder.ToList())
        {
            if (pair.Key == CurrentPlayerInitiative && iteration == 0)
            {
                CurrentPlayer = pair.Value;

                int tempHP = GameObject.Find(CurrentPlayer).GetComponent<PlayerController>().CurrentHP;
                int tempMP = GameObject.Find(CurrentPlayer).GetComponent<PlayerController>().CurrentMP;
                SetText(tempHP, tempMP);
                iteration = 1;
            }
            if (iteration == 1 && newOrder.Count != 0)
            {
                CurrentPlayerInitiative = newOrder.Max(e => e.Key);
                Debug.Log(CurrentPlayerInitiative);
                iteration = 0;
            }
        }
    }
    else
    {
        PlayerCount = 6;
        FirstTurn();
    }
}
```

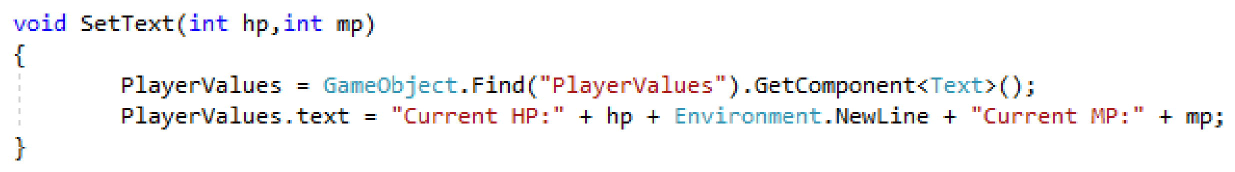
After receiving this, the EndTurn method resumes execution and checks that there are still characters taking their turn. The method then updates the “CurrentPlayerInitiative” variable by setting its value to the MAX value from list returned by the NextTurn3 method.

The following step in this process is to find the Key, Value Pair that has the same value in its Key section as the variable “CurrentPlayerInitiative”, this we can determine which character gets to go next and whose stats to update the “Current Player” panel with.

The method then goes through an “if statement” that is meant to test and see if the initiative order is changing as its supposed to.

If the “PlayerCount” is 0 then the “PlayerCount” is set to 6 and the FirstTurn method is invoked to begin the next round.

As a side note, in order to change the information shown on the Current Player panel the method SetText is used.



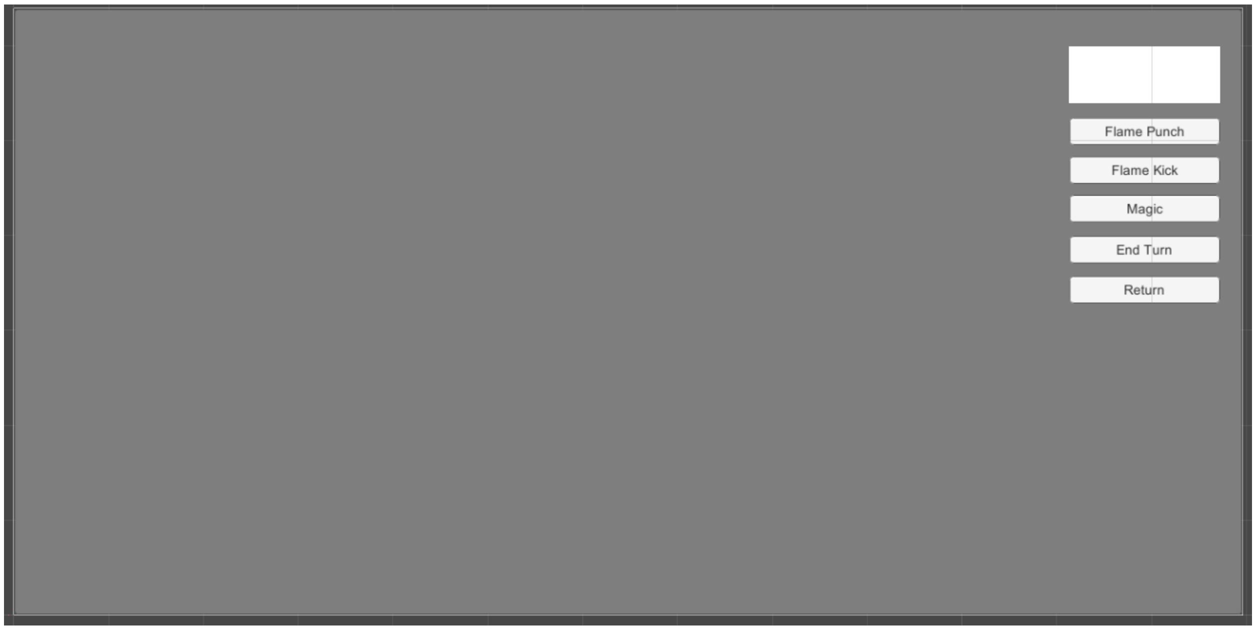
```
void SetText(int hp,int mp)
{
        PlayerValues = GameObject.Find("PlayerValues").GetComponent<Text>();
        PlayerValues.text = "Current HP:" + hp + Environment.NewLine + "Current MP:" + mp;
}
```

3.3 UI Elements and their role in the gameplay loop

When designing a game one of the most important features that need to be given serious consideration is the UI (User Interface).

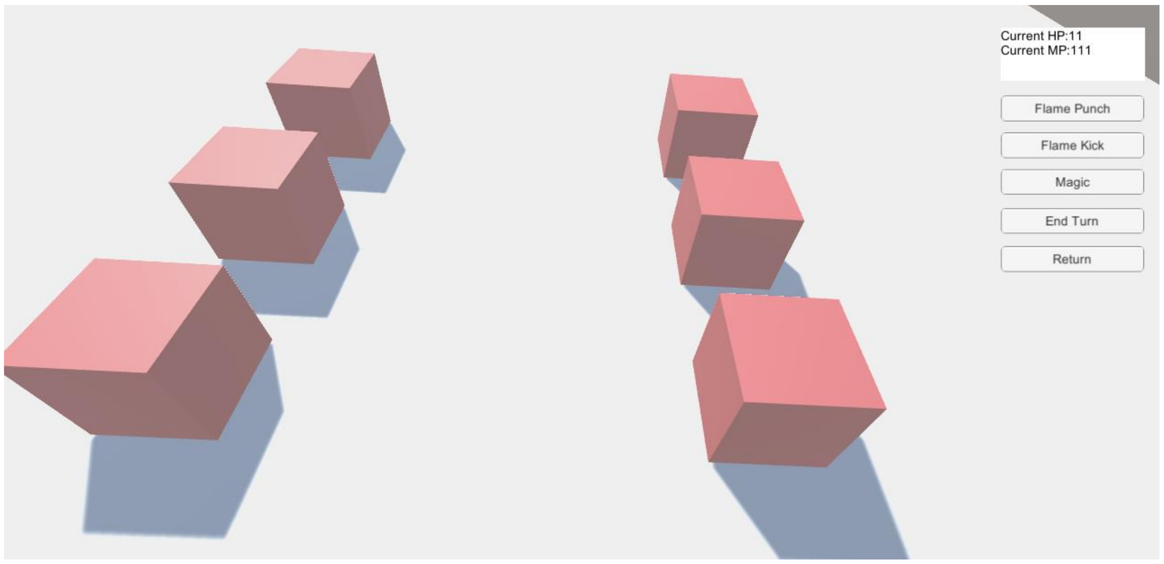
This subchapter will solely focus on this feature and its development.

Since this game was developed with mobile platforms in mind a simple UI focused on user friendliness and accessibility was the best choice.



The idea behind the UI presented above is to have as few elements on the screen that would distract the player from the gameplay and instead focus on the animations and the sound effects produced by the different attacks or spells.

The UI currently contains a “Current Player” panel which shows the stats of the character currently taking their turn, 3 different abilities, an End Turn button and a Return button that will forfeit the current match and return the player to the main menu.


Current HP:11
Current MP:111

The example above contains none of the art or animations present in the game, this is a placeholder version meant to show mechanics and UI Elements.

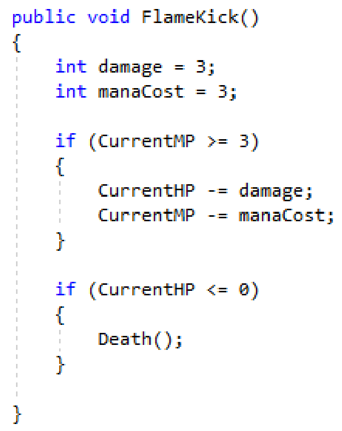
In the top right part of the screen we can see an example of the “Current Player” panel.

The End Turn button, on click, executes the EndTurn method which in turn selects the next character from the initiative order to take a turn.

The following three pictures are examples of what the code behind a damage method looks like:

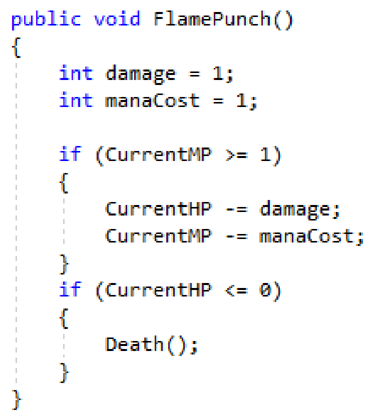
```
public void Magic()
{
    int damage = 5;
    int manaCost = 5;

    if (CurrentMP >= 5)
    {
        CurrentHP -= damage;
        CurrentMP -= manaCost;
    }

    if (CurrentHP <= 0)
    {
        Death();
    }

}
```

The Magic action is supposed to be the heavy hitter ability that deals a lot of damage but is also expensive resource wise.



```
public void FlameKick()
{
    int damage = 3;
    int manaCost = 3;

    if (CurrentMP >= 3)
    {
        CurrentHP -= damage;
        CurrentMP -= manaCost;
    }

    if (CurrentHP <= 0)
    {
        Death();
    }

}
```

The Flame Kick action is supposed to be the average damage, average cost ability.



```
public void FlamePunch()
{
    int damage = 1;
    int manaCost = 1;

    if (CurrentMP >= 1)
    {
        CurrentHP -= damage;
        CurrentMP -= manaCost;
    }
    if (CurrentHP <= 0)
    {
        Death();
    }
}
```

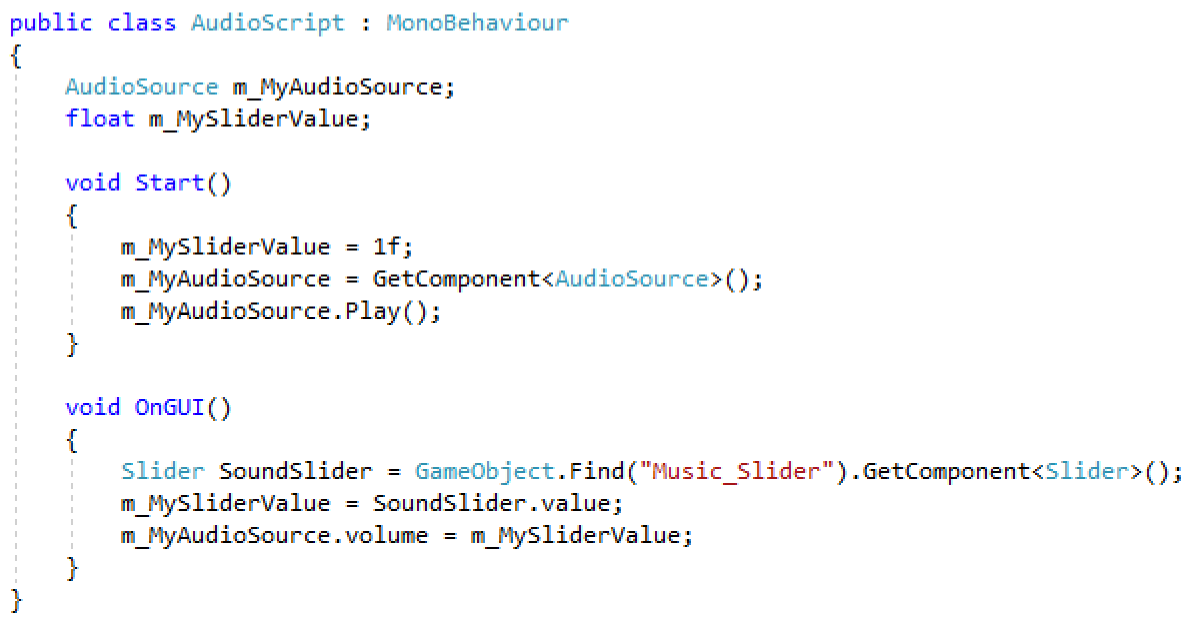
Lastly, the Flame Punch action is supposed to be the cheapest option that deals a small amount of damage.

3.4 Adding music to the game

Another important aspect of a games experience is the soundtrack. Due to financial constraints, as of the writing of this report, I was not able to hire anyone to create a theme song for this specific game. To circumvent this problem I added placeholder music. The volume of this song can be controlled in the Options Menu.

The script below creates a Slider on the Music button in the Options Menu.

This Slider is controllable and changes the volume of the Audio Source.



```
public class AudioScript : MonoBehaviour
{
    AudioSource m_MyAudioSource;
    float m_MySliderValue;

    void Start()
    {
        m_MySliderValue = 1f;
        m_MyAudioSource = GetComponent<AudioSource>();
        m_MyAudioSource.Play();
    }

    void OnGUI()
    {
        Slider SoundSlider = GameObject.Find("Music_Slider").GetComponent<Slider>();
        m_MySliderValue = SoundSlider.value;
        m_MyAudioSource.volume = m_MySliderValue;
    }
}
```

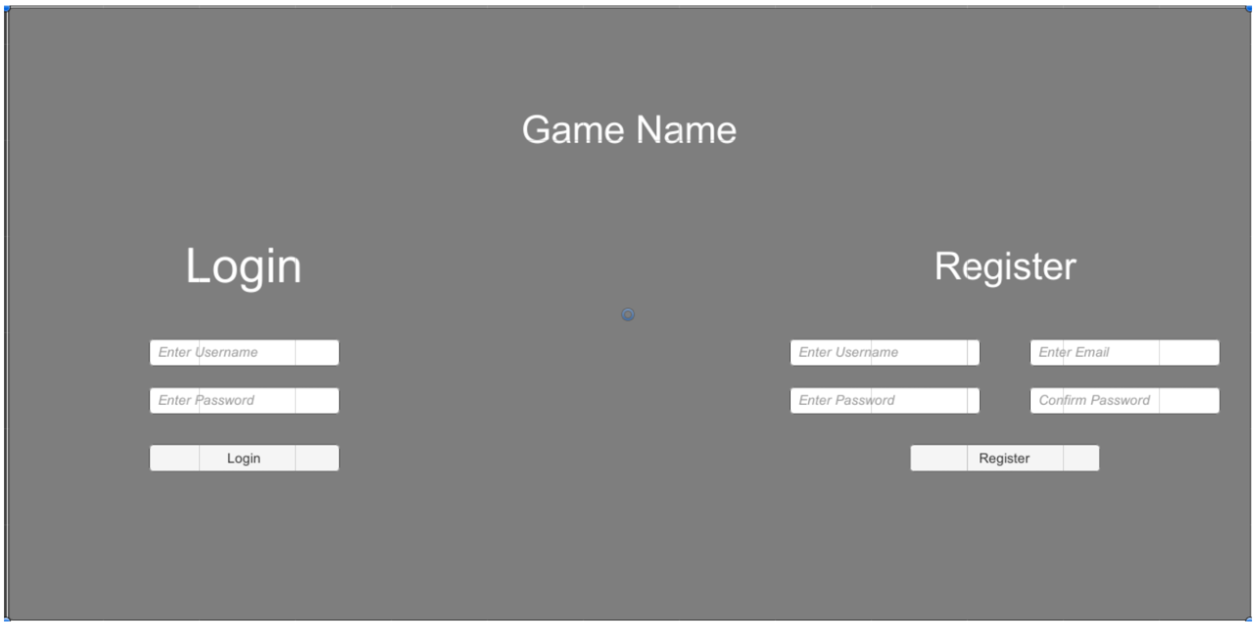
The same process is applied when adding sound effects to character animations.

3.5 Login and register (C# & Unity & .Net & SQL)

The first thing that a new player is asked to do when they open the game is to register an account. This is done to save the player data on an online database and allow them to play the game using their specific unlock on any phone as long as they have access to their account.

Part of the process is present in 3 different components of the project:

- The Game itself – UI for Login and Registering
- The Database – that houses the data related to each individual player
- The Web Server – offers a secure connection between the game and database through the use of an API



To register an account the user must decide on a username, an email address and a password. Then for security reasons re-enter that password and press register.

To login into an account the user has to enter their username and password and press Login.

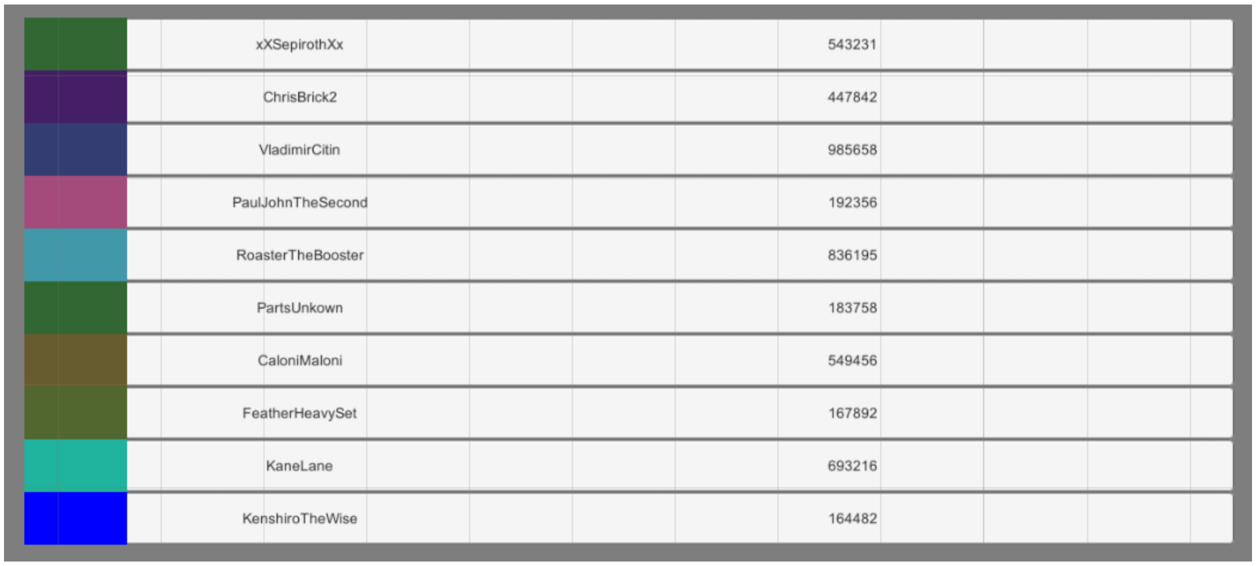
Both the Login and Register button have scripts that will make requests to the web server that will then retrieve and return the data from the database.

3.6 Leaderboard (C# & Unity & SQL & .Net)

To help breed a spirit of competition over the gameplay of this game, the idea of implementing a leaderboard came up. By having a way of recognizing the players who play the most and those who are the best at the game this style of game can reach its true potential.

The picture below is just a rough sketch of what the leaderboard looked like in the beginning.

The concept of a leaderboard is too the point and simple, promote those who are excelling at the game by having their avatar picture, name and score present for everyone that plays this game to see.



🟩	xXSepirothXx	543231
🟪	ChrisBrick2	447842
🟦	VladimirCitin	985658
🟪	PaulJohnTheSecond	192356
🟦	RoasterTheBooster	836195
🟩	PartsUnkown	183758
🟫	CaloniMaloni	549456
🟩	FeatherHeavySet	167892
🟩	KaneLane	693216
🟦	KenshiroTheWise	164482

The leaderboard makes a request to the web server, which then makes a request to the database for the high scores of all players.

During the development process the web server and the database were hosted locally and since the number of database entries and server request were relatively low compared to what a proper release on Google Play would mean, I believed that for that situation the local option was enough.

With all that said, for a proper release on Google Play or Steam or other publishing sectors this game would require proper servers that I can't currently provide.

Having a cloud server would also solve a lot of the security problems, as the companies offering these services such as Microsoft or Amazon have a reputation to uphold and secure their services.

In order to save on performance and time, the best solution for a leaderboard was to use the web server to host a one page website that showed a leaderboard that placed the top 20 player`s high scores.

The leaderboard on that website was populated using the information available in the database in the Highscore table through a web request.

In this situation the only thing that Unity needs to do is make a GET request to the web server and retrieve the leaderboard when the Leaderboard button in the Challenge Player menu is clicked.

3.7 Web server (C# & .Net & HTML)

In previous sections of this chapter the web server has been mentioned multiple times. The web server was used as a Middle Man to create a secure connection between the Game and the database. I wish to reiterate the point that not having this Middle Man would have left the database open to security exploits and would have possibly ruined the gameplay experience of many other people.

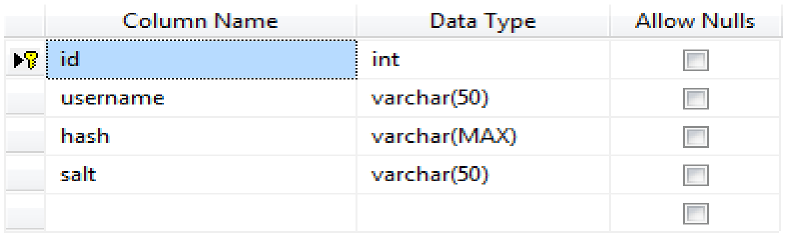
To recap, this server was used for the following:

- Middle Man between the Game and the Database
- Hosting a one page website that contains a leaderboard
- Connector that allows a secure procedure for the Log In and Register process
-

3.8 Database (SQL & .NET)

The database tables were created in SSMS (SQL Server Management Studio) using SQL.

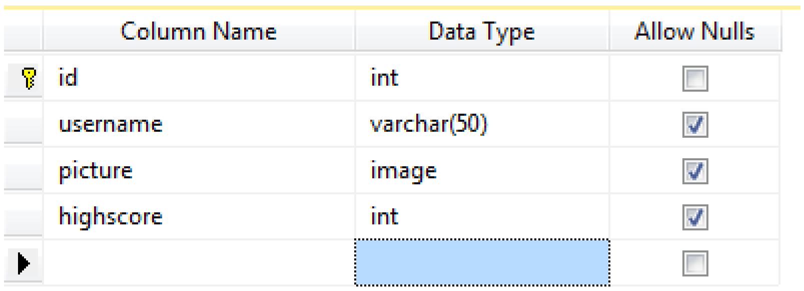
The database contained 2 main tables, one for the Login information and one for the leaderboard



Column Name	Data Type	Allow Nulls
id	int	☐
username	varchar(50)	☐
hash	varchar(MAX)	☐
salt	varchar(50)	☐
		☐

The Login table has 4 columns; id, username, hash and salt. The id field is the unique identifier that acts as the primary key in this table.

The username field holds the username of each player that has an account created. The hash field holds the hashed password while the salt field holds the salt used in the encryption process.



Column Name	Data Type	Allow Nulls
id	int	☐
username	varchar(50)	✓
picture	image	✓
highscore	int	✓
		☐

The UserInfo table has 4 columns; id, username, picture and highscore.

The id field is the unique identifier that acts as the primary key in this table. The username field holds the username of each player that has an account created. The picture field holds the avatar picture of each player. The highscore field holds each players highscore.

4. Implementation of Security Features

This chapter will focus on the changes made to features and section of the project as a whole to improve security.

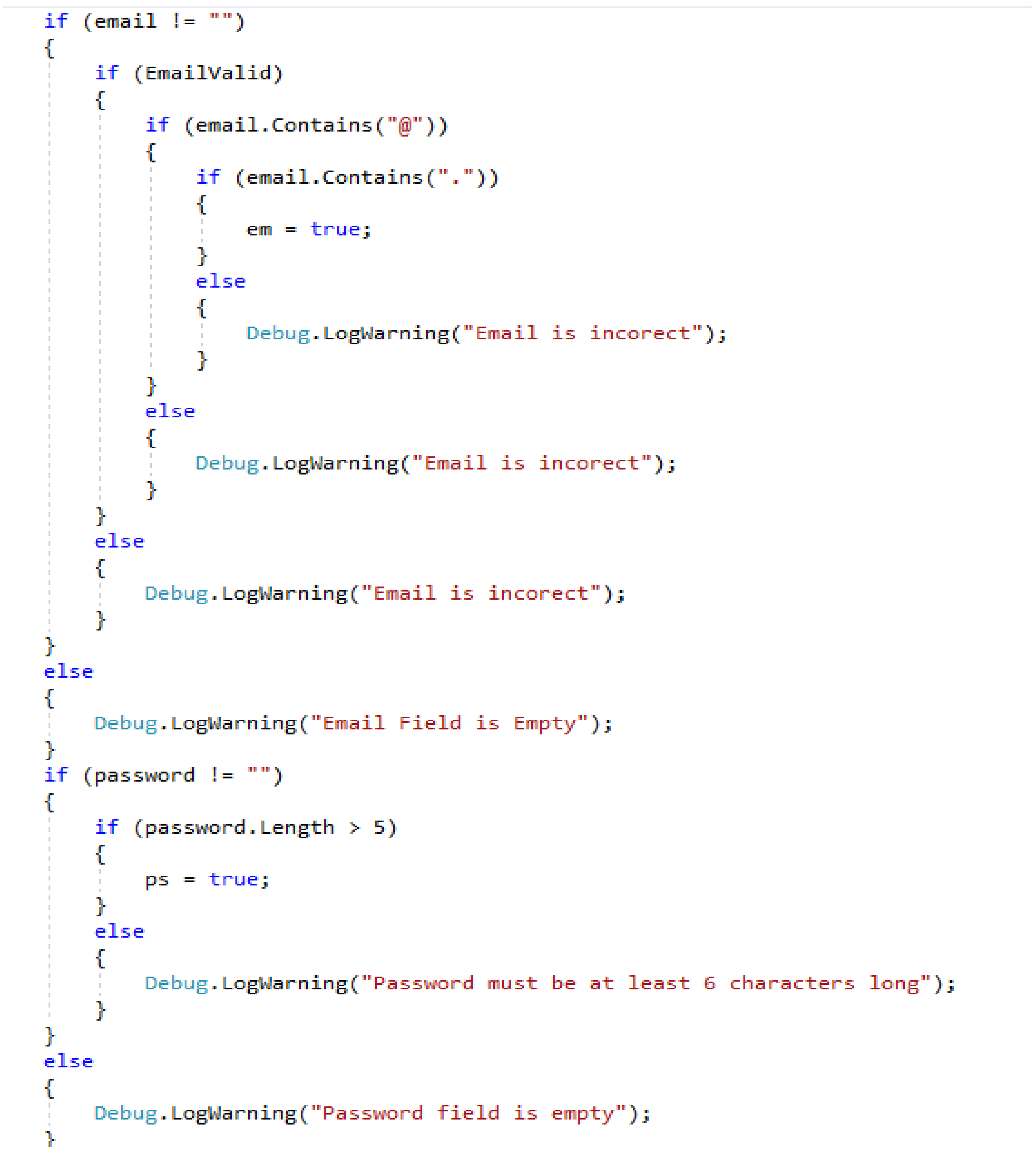
These are the main sectors were due to security changes were made:

- Login & Register – Unity
- Database
- Web server

Unity itself focuses on the development of games, in cases such as these were an external database is involved, design choices must be made to accommodate new systems that might be brought in to help offer a cheat free experience for all players.

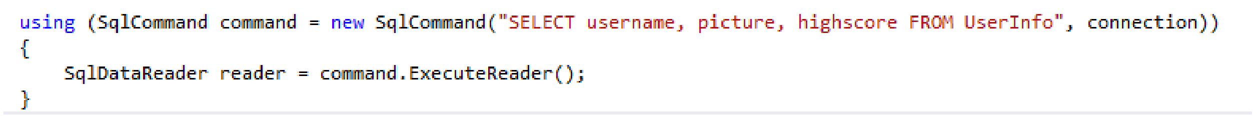
The three sectors above will be analyzed and discussed based on what changes were made or things were added to improve the security of each of them.

In the case of the Unity Login & Register sections there were added checks for the sizes and the characters that can be entered in the Input Fields provided for the Login or Register Operations.



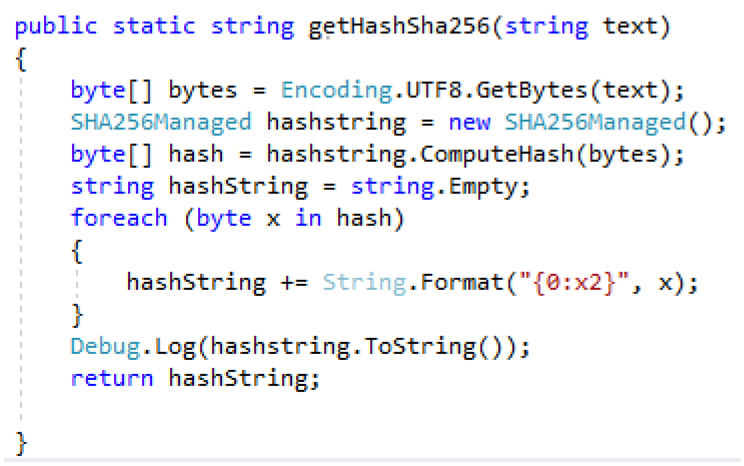
```
if (email != "")
{
    if (EmailValid)
    {
        if (email.Contains("@"))
        {
            if (email.Contains("."))
            {
                em = true;
            }
            else
            {
                Debug.LogWarning("Email is incorect");
            }
        }
        else
        {
            Debug.LogWarning("Email is incorect");
        }
    }
    else
    {
        Debug.LogWarning("Email is incorect");
    }
}
else
{
    Debug.LogWarning("Email Field is Empty");
}
if (password != "")
{
    if (password.Length > 5)
    {
        ps = true;
    }
    else
    {
        Debug.LogWarning("Password must be at least 6 characters long");
    }
}
else
{
    Debug.LogWarning("Password field is empty");
}
```

A second security measure was the use of the SqlCommand interface which helped protect our Input Fields from Sql Injections.



```
using (SqlCommand command = new SqlCommand("SELECT username, picture, highscore FROM UserInfo", connection))
{
    SqlDataReader reader = command.ExecuteReader();
}
```

The third addition is the creation of a hashing method that would encrypt the passwords during the registration and login process.



```
public static string getHashSha256(string text)
{
    byte[] bytes = Encoding.UTF8.GetBytes(text);
    SHA256Managed hashstring = new SHA256Managed();
    byte[] hash = hashstring.ComputeHash(bytes);
    string hashString = string.Empty;
    foreach (byte x in hash)
    {
        hashString += String.Format("{0:x2}", x);
    }
    Debug.Log(hashstring.ToString());
    return hashString;

}
```

The final addition from the Unity perspective is the adding of a salt. This salt is a random number added to the beginning of the hash of the password that was just entered.

For example if we have the variables of Salt = 123 and Password = cat the salt is attached to the beginning of the password (123cat) after which this new string is hashed by using the hashing method shown above.

When considering the web servers security the concepts of XSS and CSRF come to mind.

As a first step, the web server is only capable of using HttpOnly cookies which should prevent the use of JavaScript in unintended ways.

A second measure was to encode and escape any data that came from or went to the Database or the Client.

From the point of view of preventing CSRF, according to the theory, a developer should not use GET but instead use POST to transfer whatever data they have to.

One final mention in the security for Servers category is the subject of Certificates. In order for a website to be trustworthy it needs to have that green lock in the upper left side of the screen, which is why according to advice online the best choice would have been to create and sign my own certificate.

The final section for our security discussion is the database. While researching things I could do to improve the security of the database I noticed that most of major advice had already been applied in the previous sections, such as encryption (hashing + salting), using a web server as a Middle Man, making sure that all input fields are protected against SQL Injection, encoding and escaping any special characters from my database strings and server GET/POST calls and finally adding password and username rules.

Due to this the only other security improvement that comes to mind is the setting up of special roles in the database.

This is in reference to having to types of accounts; admin and guests.

The guests have only read privileges, which in this case would resume to stats request or login attempt. The admin would have full privileges and control over the database.

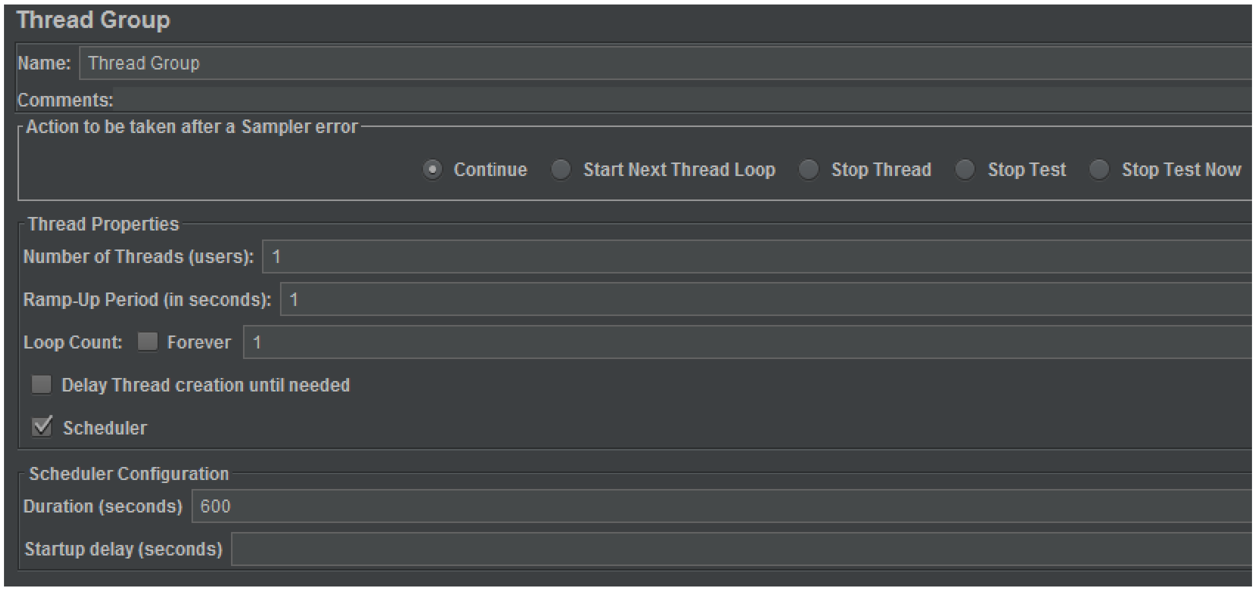
5. Software Testing and its results

The first part of this chapter will be focusing on Performance Tests Followed by Unit Test and Integration Tests.

There are 4 types of performance tests; Soak, Spike, Load and Stress.

The Soak test involves testing a product by letting it run for a long period of time at a normal, expected production load.

For this test both the Game and the Web server were tested. First of we have the web server which according to the data that was recorded by JMeter with the settings of 1 thread, 1 second ramp up time, loop count forever and the duration of this test being 600 seconds(10 minutes).



Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: Forever 1

Delay Thread creation until needed

[✓ Scheduler]

Scheduler Configuration

Duration (seconds) 600

Startup delay (seconds)

No abnormalities were detected all GET requests were answered.

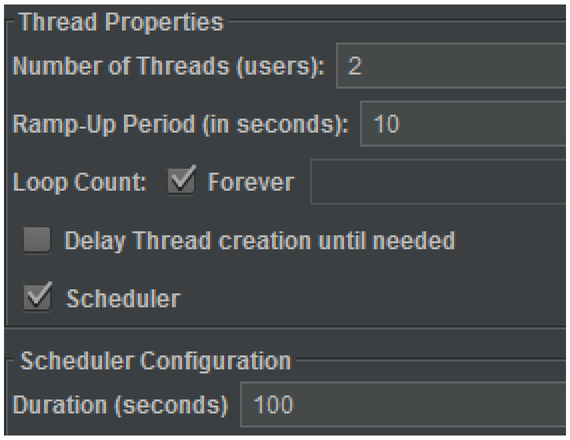
The game ran for 30 minutes without any fps losses or excessive batter drain.

The second performance test is the Spike test. The purpose of this test is to verify that a system can still function properly in case of a sudden huge wave of users.

To test this situation I used JMeter again this time using 2 thread groups.

The first thread group has the following settings; 1 thread, 1 second ramp up time, loop count forever and the duration of this test being 150 seconds while the second thread group has 2 thread, 10 second ramp up time, loop count forever and the duration of this test being 100 seconds.

In order to have a proper spike test, the first thread group was called first, followed by the second thread group and then followed by first thread group being called again.


Thread Properties
Number of Threads (users): 2
Ramp-Up Period (in seconds): 10
Loop Count: [✓ Forever]
Delay Thread creation until needed
[✓ Scheduler]
Scheduler Configuration
Duration (seconds) 100

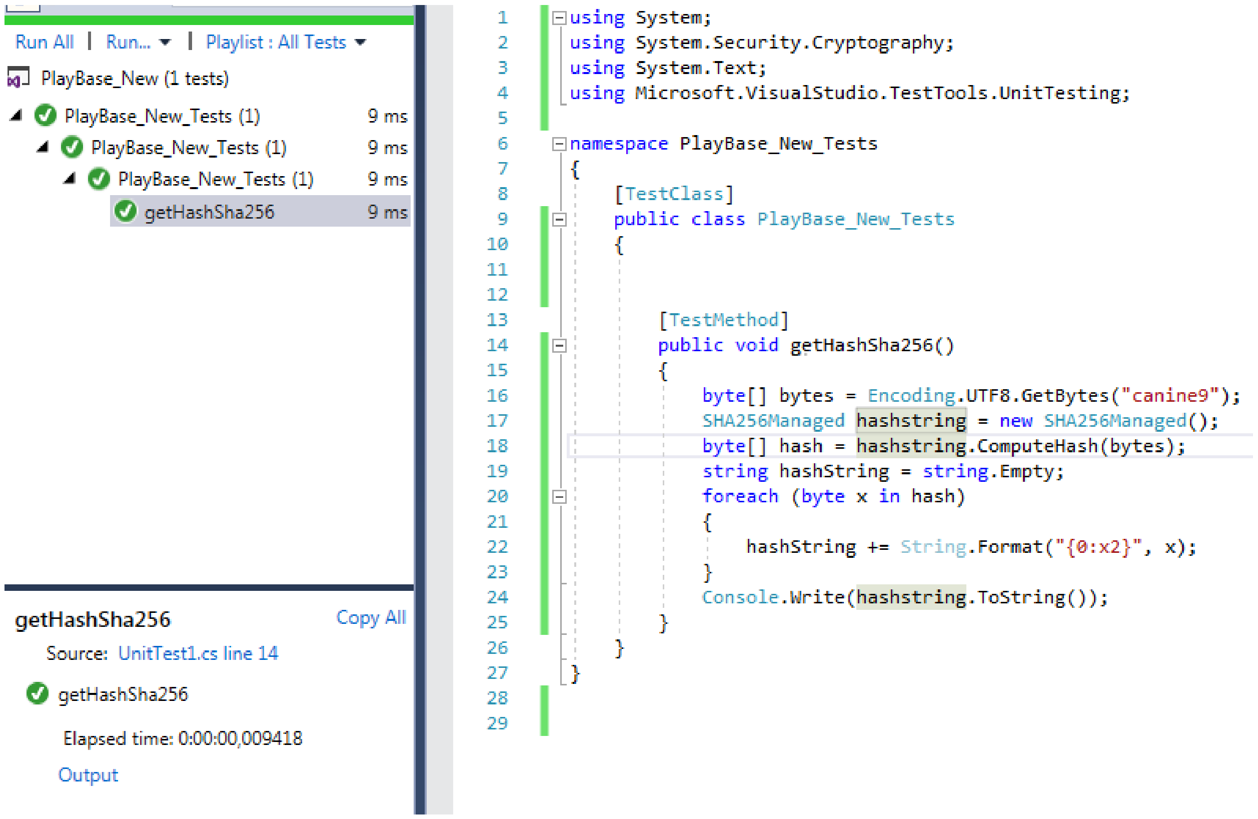
The test results showed a 0.2 slowdown during the spike phase.

The third performance test is the Load test. In this case 3 threads were used with a ramp up time of 6 minutes after which it drops slowly to 0 during the course of 1 minute. The Load test proves that this web server is capable of maintaining a decent response time of 100 ms.

For the fourth and final performance test, the Stress test a thread group of 5 threads with a ramp up time of 60 seconds and a duration of 200 seconds followed by a descend to 0 users over the course of 50 seconds. The longest response time was at the peak of 5 thread group with a time of 150 ms.

Unit Test

While doing research about Unit Testing I decided to use this method of creating tests to check if my Hashing algorithm was working properly. To my surprise there were no errors detected.



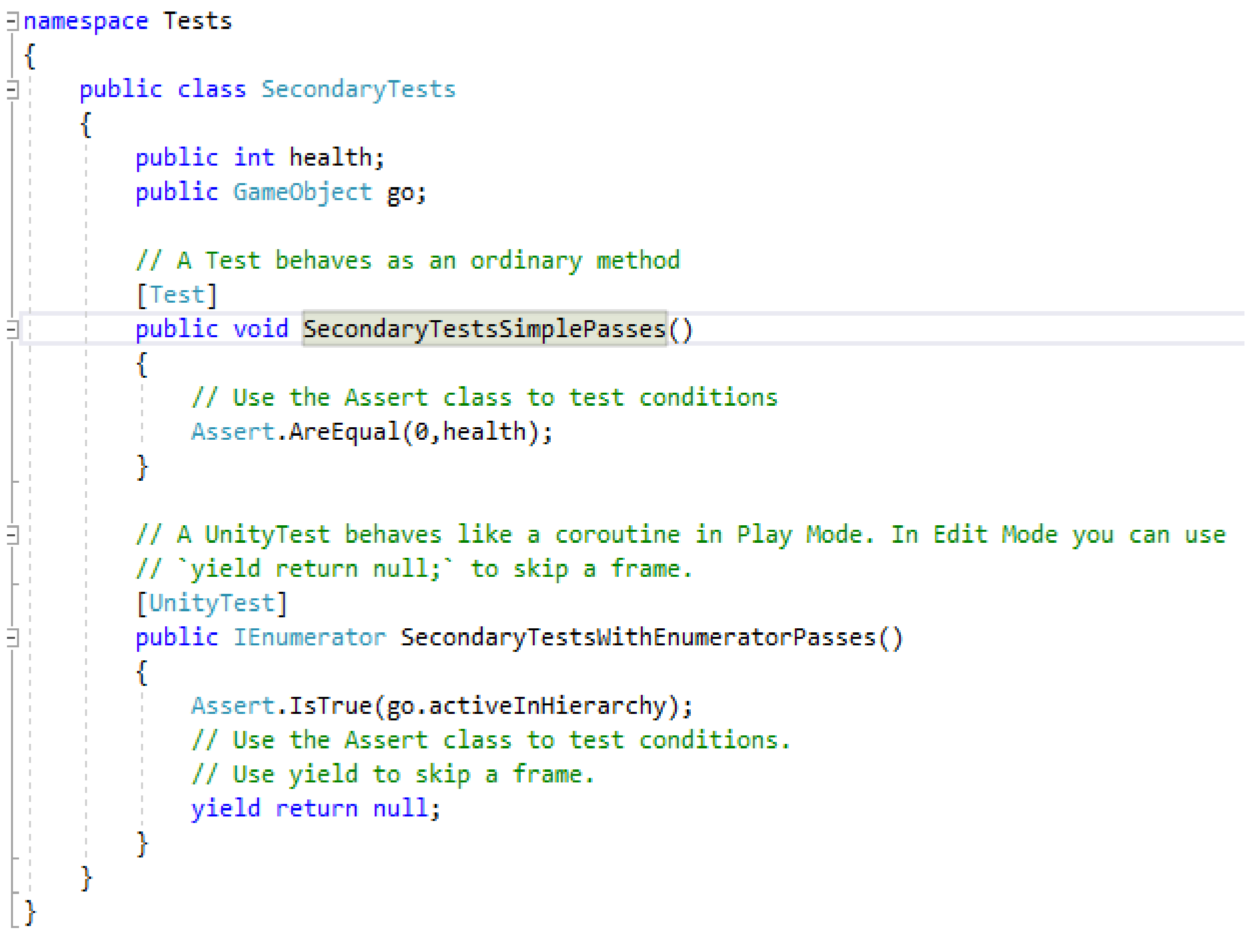
```
PlayBase_New (1 tests)
  PlayBase_New_Tests (1)            9 ms
    PlayBase_New_Tests (1)          9 ms
      PlayBase_New_Tests (1)        9 ms
        getHashSha256               9 ms

getHashSha256                       Copy All
   Source: UnitTest1.cs line 14
  getHashSha256
    Elapsed time: 0:00:00,009418
    Output
```

```
using System;
using System.Security.Cryptography;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace PlayBase_New_Tests
{
    [TestClass]
    public class PlayBase_New_Tests
    {


        [TestMethod]
        public void getHashSha256()
        {
            byte[] bytes = Encoding.UTF8.GetBytes("canine9");
            SHA256Managed hashstring = new SHA256Managed();
            byte[] hash = hashstring.ComputeHash(bytes);
            string hashString = string.Empty;
            foreach (byte x in hash)
            {
                hashString += String.Format("{0:x2}", x);
            }
            Console.Write(hashstring.ToString());
        }
    }
}
```

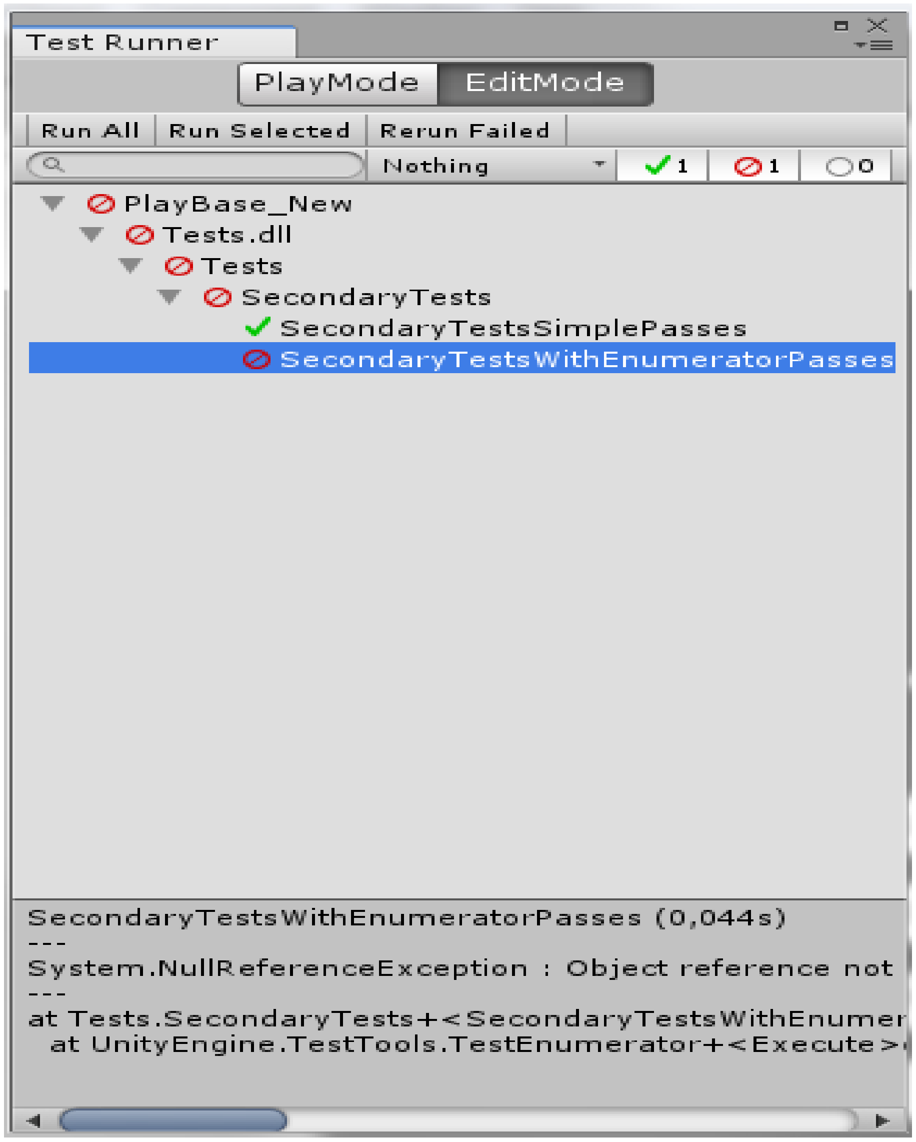
Unit tests are useful and help prevent or find errors that otherwise you might have missed. While I agree with that statement, in this case most of the methods used by the game also require the use of one or more game element provided by the Unity Engine. Unit tests require to be given a clear set of values with which to run the tests you create. In my case, for now, the Unit Test for the hashing method is the only that can be run.

Integration Test

According to information found online about Integration Tests – "Integration testing may be done at a user interface level (via automated functional tests - AFT)." [9] Thankfully, the Unity Editor comes equipped with a feature called "Test Runner" which allows developers to make simple test that involve UI and Game elements during "Play Mode" and "Edit Mode".



```
namespace Tests
{
    public class SecondaryTests
    {
        public int health;
        public GameObject go;

        // A Test behaves as an ordinary method
        [Test]
        public void SecondaryTestsSimplePasses()
        {
            // Use the Assert class to test conditions
            Assert.AreEqual(0,health);
        }

        // A UnityTest behaves like a coroutine in Play Mode. In Edit Mode you can use
        // `yield return null;` to skip a frame.
        [UnityTest]
        public IEnumerator SecondaryTestsWithEnumeratorPasses()
        {
            Assert.IsTrue(go.activeInHierarchy);
            // Use the Assert class to test conditions.
            // Use yield to skip a frame.
            yield return null;
        }
    }
}
```



```
PlayBase_New
  Tests.dll
    Tests
      SecondaryTests
        SecondaryTestsSimplePasses
        SecondaryTestsWithEnumeratorPasses
```

```
SecondaryTestsWithEnumeratorPasses (0,044s)
---
System.NullReferenceException : Object reference not
---
at Tests.SecondaryTests+<SecondaryTestsWithEnumer
  at UnityEngine.TestTools.TestEnumerator+<Execute>
```

The Test Runner Is capable of running a myriad of tests at the same time and showing you which test components have failed and where.

6. Conclusion and remarks

There are many things that I would have done differently if when I began working on this project I knew what I know now. This entire chapter is dedicated to talking about what I could have done differently and what is at this state the finished product.

1. Future Iterations and Improvements

One of the first things that come to mind is replacing the current web server with a paid server from Amazon or Microsoft, followed by placing the database in the cloud, this way avoiding a lot of security issues.

Adding voice commands that allow you to activate abilities, call out the target you want to attack instead of taping on the screen, giving you a feeling of being a commander of a small special-ops type team.

Also redesign the backend to support multiplayer properly, since many of the feature implemented this time around were made by a novice in many categories.

Getting a soundtracks composed specifically for this game, while also buying or creating amazing 3D models that can interact through different and unique animation.

One more thing, I should have added a tutorial to explain the mechanics, no matter how simple they may be.

To rephrase the last 5 chunks of text; I would, on my own time try to remake this game from the ground up and use the knowledge I gained from this experience to become a better developer.

2. Conclusion

This report covers all necessary subjects covered in Software Development TOP UP.

The subjects that this project covered are:

- Databases – the game has a login and leaderboard system that uses a database
- Software Testing – the report covers all the relevant tests that could be applied to this project
- System Integration – the use of the Web Server as a Middle Man between the game and the database
- Development of Large System – An ambition game project support by a web server and a database
- Web Security – Hashing and salting and many other security focused changes
- HCI (Human Computer Interaction) – Usability and UX Testing

To summarize this report from a business point of view, the most profitable genre of game at this moment is Turn Based Strategy, having a Login system and player Leaderboard will cause maybe people to stick with this game for a long time.

To summarize this report from a developer point of view, the main game mechanics were implemented, a database that holds players accounts and saves their progress exists and finally a server connects the game and the database to offer a secure and fun product.

I truly believe I made the right choice in picking this project for my final exam.

List of references

[1] - https://www.techopedia.com/definition/26272/c-sharp

[2] - https://i.ytimg.com/vi/B9_z0HWZ--w/maxresdefault.jpg

[3] - https://s.pacn.ws/gallery/large/GA.03979.0001.jpg

[4] - https://www.mobygames.com/images/shots/l/858795-deus-ex-mankind-divided-windows-screenshot-main-menu.jpg

[5] - https://www.overclock3d.net/gfx/articles/2016/08/18042026683l.jpg

[6] - https://i1.wp.com/windows8freeware.com/wp-content/uploads/2015/03/Unblock-Me-Main-Menu.png

[7] - https://cdn.wccftech.com/wp-content/uploads/2018/04/Graphics-settings-set-to-high-PUBG-Mobile.png

[8] - https://www.telegraph.co.uk/technology/mobile-phones/11646593/7-in-10-of-worlds-population-using-smartphones-by-2020.html

[9] - https://stackoverflow.com/questions/849467/whats-a-great-way-to-perfom-integration-testing