Volniir

A 2D RPG Rogue Like Game

Thomas Robert Barnes - S013138

Rochdale Sixth Form College  [Company address]

Contents

[1. Analysis 4](#_Toc509576198)

[1.1 Problem Identification 4](#_Toc509576199)

[1.1.1 Problem Area 4](#_Toc509576200)

[1.1.2 Computational Approach 5](#_Toc509576201)

[1.2 Stake Holders 6](#_Toc509576202)

[Stakeholder 6](#_Toc509576203)

[Description 6](#_Toc509576204)

[Use 6](#_Toc509576205)

[Computer Skills 7](#_Toc509576206)

[1.3 Research 7](#_Toc509576207)

[1.3.1 Binding of Isaac Rebirth 7](#_Toc509576208)

[1.3.2 The Elder Scrolls V: Skyrim 10](#_Toc509576209)

[1.3.4 Existing Solutions Summary 11](#_Toc509576210)

[Primary Research 11](#_Toc509576211)

[Questionnaire 11](#_Toc509576212)

[1.3.4 Primary Research Summery 12](#_Toc509576213)

[1.3.5 Essential Features 12](#_Toc509576214)

[Possible Solutions 13](#_Toc509576215)

[Scratch 13](#_Toc509576216)

[Unity 13](#_Toc509576217)

[1.4 Proposed Solution 14](#_Toc509576218)

[1.4.1 Visual C# 14](#_Toc509576219)

[Hardware and Software requirements 14](#_Toc509576220)

[1.4.2 Objectives and Success Criteria 16](#_Toc509576221)

[Limitations 19](#_Toc509576222)

[Interview Evidence 20](#_Toc509576223)

[2.Design 34](#_Toc509576224)

[2.1.1 Modular Design 34](#_Toc509576225)

[2.1.2 Modular Design Explanation 35](#_Toc509576226)

[Class association 35](#_Toc509576227)

[Class interaction 36](#_Toc509576228)

[2.2 The Solution 37](#_Toc509576229)

[2.2.1 Data Structure 37](#_Toc509576230)

[Game Data 37](#_Toc509576231)

[User Interface Designs 38](#_Toc509576232)

[Introduction Screen 38](#_Toc509576233)

[Leader boards 39](#_Toc509576234)

[Main Menu 41](#_Toc509576235)

[Start Game 42](#_Toc509576236)

[Exit Game 43](#_Toc509576237)

[Game Screen 44](#_Toc509576238)

[Testing Strategy 46](#_Toc509576239)

[Testing Strategy Description 46](#_Toc509576240)

[System Test Plan 46](#_Toc509576241)

[Development 48](#_Toc509576242)

[Prototype 1 48](#_Toc509576243)

[Character movement code 48](#_Toc509576244)

[Prototype 2 49](#_Toc509576245)

[Character movement code 49](#_Toc509576246)

[Prototype 3 50](#_Toc509576247)

[Character movement code 50](#_Toc509576248)

[Technical Documentation 51](#_Toc509576249)

[Class Association 51](#_Toc509576250)

[Dependency 51](#_Toc509576251)

[Tears & GameDisplay 51](#_Toc509576252)

[Enemy & GameDisplay & Game1 53](#_Toc509576253)

[CharacterSelectionDisplay & GameDisplay 55](#_Toc509576254)

[MainMenuDisplay & Game1 56](#_Toc509576255)

[LoadingScreenDisplay & Game1 56](#_Toc509576256)

[Aggregation 57](#_Toc509576257)

[Rocks & GameDisplay 57](#_Toc509576258)

[Items & GameDisplay 57](#_Toc509576259)

[Entities & GameDisplay 58](#_Toc509576260)

[Tears & GameDisplay 58](#_Toc509576261)

[Composition 59](#_Toc509576262)

[Rocks & GameDisplay 59](#_Toc509576263)

[Enemies & GameDisplay 59](#_Toc509576264)

[Soundtrack 60](#_Toc509576265)

[Algorithm 60](#_Toc509576266)

[Leaderboards Screen 62](#_Toc509576267)

[Binary Reader 62](#_Toc509576268)

[Binary Writer 64](#_Toc509576269)

[Collisions 65](#_Toc509576270)

[Rocks 65](#_Toc509576271)

[Enemy and Player Projectiles 67](#_Toc509576272)

[Testing 68](#_Toc509576273)

[Proposed testing 68](#_Toc509576274)

[Testing Table 68](#_Toc509576275)

[Testing Video 70](#_Toc509576276)

COMP 3

# 1. Analysis

## 1.1 Problem Identification

The project is going to be a 2D Dungeon Crawler RPG game that follows a knight travelling through a cursed land. The project follows a detailed lore that is based in Oslo Norway; the lore is composed of true events that happened in the 1990’s. The project will include a fully designed combat feature, looting feature and a crafting feature. It will be used as a marketing campaign/tool to promote the new EP by the band Volniir called “The Enigma” and will be inspired by the stories told in that EP.

The project will include a few fixed rules for the computer to follow as the game is played. There will be rules for the computer to manage the Health of a character as the game is played.

The computer will also handle the experience points of a player as they progress through each level killing enemies there will be calculations, like addition, to total up the XP gained by a player, the experience gained could be quite large so it would be more efficient for a computer to handle it.

A mana feature will be implemented if the player uses magic based weapons; this would be handled by the computer, as mana would be deducted and regenerate over time. A computer would handle this more efficiently because the mana would have to decrease by a certain amount each time but also regenerate by a certain amount at the same time. On paper, this would require many calculations at once and would take up a lot of space, whereas a computer would update a variable using less memory and not having to create a new variable each time.

### 1.1.1 Problem Area

The project Volniir has been inspired by many games released since the 1980’s to present day, as the Role Playing Game genre was introduced by the game Final Fantasy developed in japan by Hironobu Sakaguchi. The genre lets players connect with the games characters and lets them feel what the character in the game feels; it lets them experience fantasies they never would experience in real life. The games make the impossible seem possible with integrating the player into the world generated by the game.

In 2011 the game Skyrim (the 5th game in the elder scrolls series) was released by Bethesda games, it introduced the mainstream media into the world of Skyrim infested with dragons, trolls, dwarfs, elves, Nordic beings and other magical creatures.

Other indie games like Fez and The Binding of Isaac are influences of the project, the art style of Fez using vibrant colours to represent a beautiful and alive environment, transitioning to dull and grey colours to represent a sense of death and despair.

The game Binding of Isaac gave publicity to the dungeon crawler genre of games, the basic principle was that every time you died you lost all progress of each level forcing the player to restart the game with a fresh character.

Another game, the project is inspired by, is DOOM the 1993 first person shooter, this game introduced state of the art 3D graphics and a fast-paced movement system to gamers; the main focus was to only shoot demons and survive

The main idea of my project is to use elements of the RPG genre to entice the player and let them get lost in the environment of story what surrounds the player and to use elements of dungeon crawlers to increase the difficulty making each decision the player makes crucial to if the player wins or loses. The project is needed to help innovate the RPG genre by adding extra features to give gamers a new experience for other future games, by other developers, to be inspired by.

### 1.1.2 Computational Approach

The game will save after each level or at a pre-set point through the game the characters data (health, XP, inventory, mana) would be stored in an encrypted separate file that is unique to the user, therefore a user can continue their game and \*have their save secure. A calculation could be used to countdown the time remaining before each save, and will save this to a single variable taking up less space as it would on paper.

The computer will also store the rules for inventory space of a character, each item could weigh a certain amount and the computer adds all of the amounts together but limits the carry amount to a certain value.

The project will also include a login feature that is secure and encrypted to stop players from modifying save files; the computer would handle this by using a username and password system to stop a user accessing another user’s save file.

The game will require a User interface to display features like the players health as they progress, as well as an experience bar to represent their experience gained until the next character level. In addition, the user interface could display a button to exit the game or change the games’ settings.

## 1.2 Stake Holders

### Stakeholder

Volniir is a 2D RPG Dungeon crawler game, the projects intended stakeholders is that of any fan of the band volniir, gamer, a casual gamer, hard core gamer or competitive gamer to introduce them to upcoming volniir news and spread publicity. The project will be designed with an element that allows any person to play the game and enjoy it for many hours. The project’s main stakeholder would be a fan of Volniir (age 16-25) who mainly plays RPG games, as they would have more interest in the lore and overall gameplay of the project as it relates to many stories told in Volniir songs. They would play the project longer as they would have an interest to fully complete the project, discovering all the secrets, achieving max level etc. The game would enable the fans to experience Volniir in a different format and add a layer of interaction between the band and the fans, allowing them the experience the songs and themes of Volniir but as a game.

Our secondary stakeholders would be the casual gamer (age 16-32) as they would still have an interest in the project however, they may lose interest and stop playing after so many hours of gameplay due to them preferring other genres of games or not having an interest in the band.

The stakeholders who may take an interest in the project are normal working people (age 25+) as they sometimes play games after their work hours to relax.

In addition, reviewers may be interested in the game to give the game a Meta score and to give publicity to the game.

Finally, other developers will be interested in the game to take inspiration in the game’s design, game mechanics and art style to help develop their own game.

### Description

The hard-core RPG gamer would be more interested in the project’s mechanics as they would require a certain gameplay style and features that complete the RPG experience, they would play 3+ hours on the game wanting to fully complete it as being unemployed or being in education they would have more time to play games on a daily basis.

The reviewer would have more experience in the game field due to them having to review games for a living; so they would look at all mechanics in the game, the bugs of the game and the overall design of the game to give it a score. As well as gain interest into the bands songs and activities.

All stakeholders would require login information so they can access their own save files, one stakeholder could have administrator access to help revert save files or remove a user account.

No stakeholders, apart from the Admin, will have access to modify or delete save files, they also will not have access to the game’s data that stores HP, XP, Damage, item IDs so they cannot cheat their way through the game.

### Use

All of the stakeholder’s intended use of the project would be for entertainment and to pass time during travel, however the hard-core gamer may use the game for its detailed lore for inspiration and creativity, or they may want to complete the game in all ways possible. However, the reviewers intended use would be to review and analyse all aspects of the game, they would fully complete the game and possibly look into the games code. Furthermore, reviewers and game developers would look at how detailed the game is and how the game appeals to people.

### Computer Skills

The hard-core gamer as well as the reviewer and the game developers, would have more advanced computer skills to play a game because most of their time is taken up by playing games and completing them. They would also have more of an expertise in RPG games as they prefer that genre of games, therefore they would know all of the basic mechanics and strategies in beating a game. They would also know how to modify and edit computer files, also they may know how to use console commands to edit their system.

An average gamer would know how to use basic controls, locate game save files, install, and uninstall the project.

My user interface based upon my stakeholders would have to be fairly complex but easy to use, as the game would have to appeal to all people of different abilities. I would use medium sized buttons to navigate the menu, or the settings tab to edit audio levels, subtitles, or video settings. I would need a UI to display the user’s health, XP, mana, inventory space, stamina and quest information; I would also need one to display the enemy’s health and level. The UI is going to be based upon the designs of other similar games, for example a small and clear UI to use less space on the screen. The project will also have a section for user login and administrator login.

## 1.3 Research

The research for the project is crucial to it due to the project being inspired by different games and their mechanics that overall shape how the game plays, looks and feels to the player.

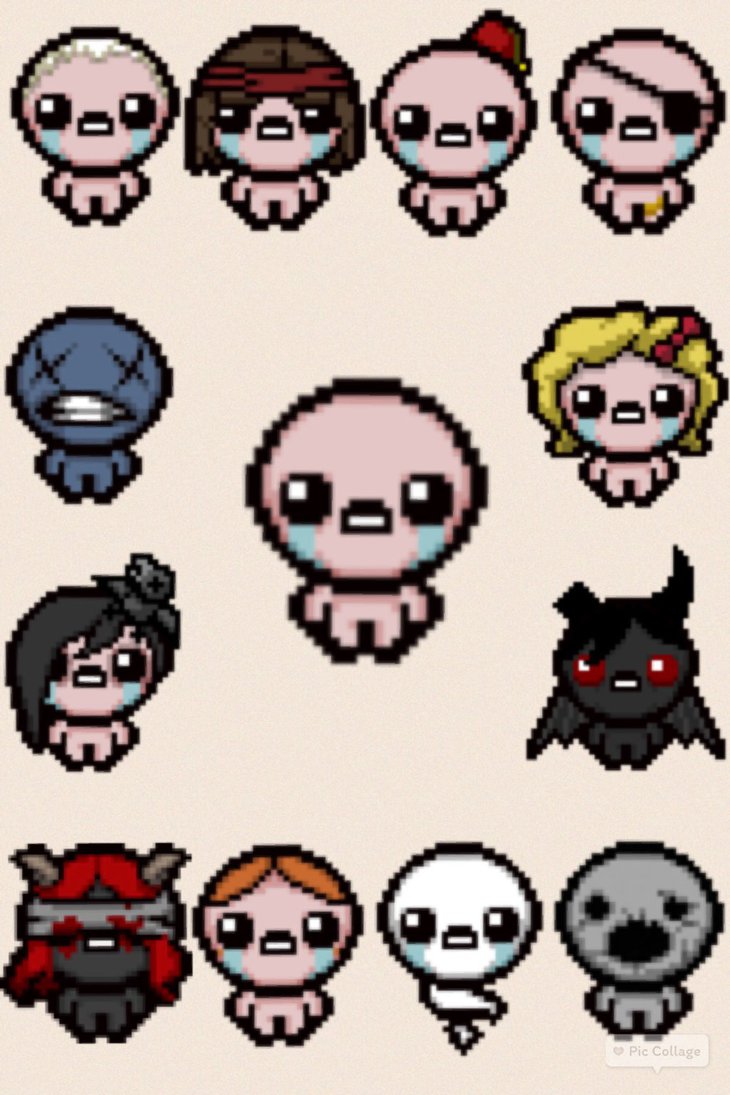
### 1.3.1 Binding of Isaac Rebirth

Binding of Isaac Rebirth is a rogue like role-playing game that is a sequel to the game Binding of Isaac, it revolves around the character Isaac and his fight in his basement against the demons and monsters trapped down there. The player spawns in “The basement” they start with two bombs, our-fixed health points and a weak projectile; they can progress by entering each neighbouring room and eliminating all enemies in the room.

If the player is hit by an enemy or damaged by explosives, they lose half of a health point that does not regenerate unless the player manages to find a health pickup, which can be very rare. At any point, if the player does die they lose all items they had and they have to restart the game all over again. This feature allows the player to play the game for over 100+hrs as the difficulty increases every level they explore but is addicting to continue and beat their last play through; however, this feature can get annoying usually ending up with the player losing interest in the game or quitting midway.

The movement system is quite hard to master, as it is a loose movement, the character does not stop when the movement key is released; this further adds to the difficulty to the game because the character could accidently hit an enemy’s projectile and lose health, which eventually could lead to death.

The game also includes a large and detailed item list that displays all the items the player can access when playing the game, it includes a description of each item and the statistics of the item. For the end of every level is a boss battle, the boss are usually large and are unique compared to the normal enemies spawned in the game. They have more health and special abilities that make them hard to beat unless you have a good item setup.

The bosses vary every level, randomised in every play through. Some bosses can be difficult compared to other entry-level bosses; for example Gurdy compared to Satan. Each boss has different phases of attack where they repeat the same attack pattern but change to another attack when they reach a certain health. As shown, Satan has a laser attack pattern that deals one heart of damage if he hits the player. This attack alternates to a circle attack rather than a linear attack, and cannot be avoided therefore the player has to take damage or die. He is the hardest boss in the game requiring the player to have gathered good items throughout the game. The player obtains good items by receiving luck, to increase their luck they have to use cards what they pick up throughout the game; these cards can increase a players luck or decrease it depending on the type of card. Their luck can also vary depending on what character they play. If the player selects, a character like Isaac their luck will start at 50%; if they play Greed their luck is decreased by 4% every floor they complete. Each character has their own ability and statistics for example, Lazarus spawns with three hearts, two damage and one bomb, furthermore when he dies he respawns with one heart and keeps all of his items. Cain spawns with two hearts, 3.5 damage and one key but he has a starting item, Lucky Foot, this item increases his luck by 25% meaning his chances of getting helpful items is increased at the start of the game.

The game has a large fan base for a three-year-old game; thousands of people watch gamers live stream the game doing speed runs of the game, trying to complete it in the fastest time possible. People also create their own modifications of the game for other people to download and play; these mods change the player’s experience.

### 1.3.2 The Elder Scrolls V: Skyrim

The Elder Scrolls V: Skyrim is an action role-playing game that was released in 2008 by Bethesda. The game introduced the new setting of Skyrim, an island that is stuck in between a civil war due to a Jarl being assassinated one year before the game is set. The player plays as the main protagonist, Laat Dovahkiin (the last Dragonborn), who is being sent to his own execution until a Dragon saves him. The game gives the player the option to customise their character, allowing them to change its race, gender, hairstyle, height, build, eye colour etc. This feature gives the player unlimited customisability, integrating the player into the world of Skyrim.

The character can be edited at any time during the play through of the game where they can go to an NPC in one of the major towns; it does cost the player to change their appearance.

The Elder Scrolls V: Skyrim does have a detailed combat system. This system is constructed from the four basic playstyles: one-handed combat, two-handed combat, long-range combat and magic combat. These playstyles are determined from the weapon the player chooses use throughout the game; the more the player kills enemies with that weapon the more experience they gain for that weapon’s combat skill tree. For example if the player uses one handed daggers for most of the game the one handed skill tree levels up, as that skill tree levels up more abilities are available for the player to unlock using skill points. These different abilities can improve the player’s experience with that combat system, for example, one-handed weapons do more damage and use less stamina. However, the player can change their preferred combat system at any time during the game.

A major drawback of the game was that upon release the game included many errors in the game that greatly affected the player’s experience of the game and required a patch to fix the game after release.

The game is set in a detailed environment, filled with life. The vegetation and forests are filled with vibrant colours and trees that can be up to 12 foot tall. They also house creatures that live in the forests for example: wolfs trolls, raiders, goblins, werewolves etc. The mountains in Skyrim are based upon the Alps in France, their peaks reaching the clouds and covered in snow. The player can encounter enemies like frost trolls and even dragons that live on the peak of the mountain, these enemies usually have special attributes that belong in that environment, for example frost damage.

The player can access all areas in Skyrim’s map, no area is blocked off to the player, giving them freedom to explore and learn about the ancient land.

Finally, the game is based around the detailed lore of the lands of Skyrim, which is hinted at in the earlier games in the Elder Scrolls series. The player is taken through the civil war between the Empire and The Stormcloaks, however Dragons start to resurrect when the player finds out that they are Dovahkiin (Dragon Born). The DovahKiin is the only Nord that can defeat truly defeat the Dragon threat as once they kill a dragon they absorb its soul fully killing them so they cannot be resurrected to terror the lands again. The outcome of the game is decided on that player’s action, if the player decides to side with the Greybeards they are influenced to learn more of the words of the Dragons and are influenced to side with the Dragon threat.

### 1.3.4 Existing Solutions Summary

For my project, I would like to combine features from both games, The Binding of Isaac and The Elder Scrolls V: Skyrim as both games have unique aspects that would benefit my project.

The aspects I would like to add to my project from The Binding of Isaac are the rogue like element of the game, where wen you die the game resets, because it increases the playtime of the project as well as add some form of difficulty of the game. I would also like to add bosses based upon the bosses in the game as it keeps the user interested in the game by challenging the user. From the Elder Scrolls V: Skyrim I would like to implement the skill tree feature where when the user reaches a certain experience level they can upgrade their skills to improve their gameplay experience. Furthermore, I would like to add the different combat styles in my project as it gives the user a choice on how they want to play and changes their experience of the project based on their decisions.

### Primary Research

### Questionnaire

A questionnaire was made for research that will help the project appeal to the intended stockholders. Sixteen questions were made and split up into sections to help gather quantitative data for the project. This questionnaire gives the project the data on what the stockholders want from the project and what should be included in the project.

The first question asked the person’s gender as it helps the project know what appeals to each gender. There were four males and one female asked to complete the questionnaire.

The second question asks how old the person is; all participants either were aged 16-20 showing that they all are in education or have a part time job; meaning that they have some free time to play.

The third section asks the participants general questions about themselves and what games they play. All five participants answered that they do play games, four participants answered that they do play Role Playing Games, three participants answered that they play rogue like games, three participants play indie games and three participants have a job.

The fourth section asks the participants questions about what gaming platform they play on. The majority of participants agree that PC and Xbox 1 are the best platforms to play on, as well as Xbox 1 being more optimised to play games on; however, the participants mainly play rogue like games on PC. Most participants play rogue like games on PC; however one of them does not play rogue like games at all.

The fifth section asks the play time (per week) of different scenarios, on average the participants play 5-6 hours a week playing games, 3-4 hours playing shooter games, 1-2hours playing RPG games, less than an hour playing rogue like games and less than an hour playing indie games.

On the final section on average, the participants rank The Elder Scrolls V being the best game on the list provided and Undertale being the worst game on the list provided.

### 1.3.4 Primary Research Summery

Summarising from the questionnaire and the interview made, the data gathered tells the project that the project should have a long playtime as the data concludes that most people play 5-6 hours a week playing games. Therefore over a month people play 24 hours, so in order for the project to be appealing the game has to have an average completion time of 12hours. The data from the interview concludes that artwork is a key point to add my game as it makes other games unique and appealing to the player, as well as the game should include features like permanent death, Challenges, Shops and bosses. The project should also be error free as possible as the participant of the interview dislikes broken games when they are released.

### 1.3.5 Essential Features

Concluding from both the primary research and secondary research the project should be based upon games like Binding of Isaac, where features like the permanent death and a fixed health system should be added as they increase the difficulty of the game and further expand the playtime of the game. It also appeals to the projects main stakeholder who plays 2D RPG games and rogue like games, therefore they would want to play the game. Also from the primary and secondary research aspects from the game The Elder Scrolls V Skyrim should be added for example the large-scale map and detailed quests as they give the game more depth and exploration. Finally, the soundtrack should be based off old 90’s 8-bit style that is minimalistic and not annoying, which gives an atmosphere to the game.

The inputs that need to be added into the program is a login system where the user enters their own email and password this is so a players progress can be saved as the playtime of the game is going to be 12hours+. Another input is the control scheme of either using WASD keys or the left joystick on a controller to move a character as in the interview the participant prefers a gamepad.

The outputs needed in the project are when the user tries to move right the character on the screen should move to the right. As well for the fixed health feature a health bar should be displayed which outputs the player’s health. Another output should be text boxes that output dialogue what NPC’s are saying to the player, this helps drive the lore of the game.

The problems in current systems like The Binding Of Isaac is that the game is broken upon release, this effects the experience the player receives therefore before the project is released all bugs should be found and fixed so the game plays as intended. This was one of the points made in the interview as the participant thinks that games are rushed and not checked over.

## Possible Solutions

### Scratch

A possible solution to make the project is to use the program scratch. To make the program pre-programmed algorithms can be dragged and dropped into different objects to create functions, for example move a character around the screen. The library of assets can be used to easily make a character to be used in the program, however the quality of the assets are very limited so high resolution assets cannot be imported into the program. Furthermore, in order for a user to play the game they will have to have scratch pre-installed in their computer system. Finally, the functions that can be made are very limited and simple so if certain physics need to be added into the game cannot be programmed into it as scratch limits the complexity of the program.

### Unity

Another possible solution is to use the game engine Unity to develop the game. The way the game could be made is to use the game engine’s toolbox to create basic shapes that can be combined to create a character or a scenery, however in order to create custom 3D shapes a separate 3D modelling software is needed to create these objects. Assets as well in unity can be high resolution (up to 4k) however, the drawbacks are that the assets have to be created separately and if you want good assets, you have to pay for them, which can become expensive just for a simple scenery.

## 1.4 Proposed Solution

### 1.4.1 Visual C#

The game project will need a main menu using buttons so the user can start the game, access tutorials and access the settings menu. C# would be ideal to create the solution for this due to C# being able to use multiple forms to create a secondary screen that can be accessed by using buttons. These buttons can be easily implemented into the program by using the simple toolbox to drag and drop buttons into the application. The button then is automatically set up with its own function, by C#, that can be edited easily and different functions can be added, or changed to suit the use of the button.

The game project will need to be able to create separate user accounts, which allows each user to save their own game progress and statistics when they play the game. C# can implement code from the library code available from visual C#. System.IO can be added to the program for file handling that writes certain data to separate files; therefore, C# could store a, name, statistics and more into unique user files that only can be accessed by the user logging onto their own account saved or playing on their local system. If they do not have an account, file handling will be used to make a new file and then the user to input their account details to be saved. C# can use validation so the email and password are needed for a user to create the account; this stops errors from occurring in the project.

Custom graphics will need to be added so the game can look appealing to users and have its own art style and character design. C# allows custom graphics to be added into the application as well as being applied buttons, by importing custom assets into the program. C# can also customise the text properties to allow the project to utilise different text fonts, sizes and properties to further add to the design of the game, custom text fonts can also be imported into the project to be used.

This can be handles by the monogame add on to Visual studio as it can handle the import of custom graphics as well as manage velocities so player movement will be easier to implement and handle.

In order to make the project the application will need to be setup and a folder structure made to store the different files of the project. C# automatically sets up the basic application and folder structure; this saves time as no manual work is needed to initially create the program.

### Hardware and Software requirements

|  |  |
| --- | --- |
| **Hardware** | **Why?** |
| Intel i3 6th generation 3.2 GHz CPU | The game will mainly run off the CPU as well as the operating system so a medium spec CPU is needed. A minimum of 3.2 GHZ is required so the game and operating system can run at fast speed as well as run off the 4cores of the intel i3 chip; (as many calculations will be made by the project |
| 6gb DDR3 RAM | The operating system needs a minimum of 2GB of ram and optimally the project will need up to 1GB of ram to operate due to data being temporarily stored when the project is running. |
| Nvidia GTX 760 GPU 2GB VRAM | The project needs a graphics card to render the graphical interface of the project as well as render the project; as well, the operating system needs the graphics card to render the operating system. |
| WD 124GB hard drive | This is needed to store the project itself and further updates that could take up to 5GB as permanent user files are going to be stored by the project and updated. Furthermore, the operating system needs the hard drive as storage space and requires it to be installed on. |
| Samsung 1080p 75hz monitor | A monitor is needed in order for the User to use the computer so they can see the desktop. As the project will be running at 60FPS, a 75Hz refresh rate monitor will be needed so the user can experience 60FPS. |
| Corsair Mechanical Keyboard | A keyboard will be needed to control the character in game and to navigate the operating system. Furthermore, a corsair keyboard will be needed to reduce Ghosting keys and so users can map keys to perform specific moves in the project. |
| Corsair Optical Mouse | A mouse is needed to move the cursor across the screen as well as control the direction of the User’s projectile in the project. An optical mouse will be needed for extra accuracy and an increased DPI. |
| **Software** |  |
| Windows 10 x64 | An operating system is needed so the whole computer can work as well as create a display for all of the components of the pc to interact with each other. Windows 10 will be needed, as it is more compatible with Visual Studio that is the base software of the project. The operating system is also more optimised for all programs as it uses fewer resources from the CPU and RAM. |
| Volniir | The game will be installed on the PC in order for Users to play the game; this will be programmed in C#. The system needs to be compatible with C# and visual studio as that is the base of the game. |

### 1.4.2 Objectives and Success Criteria

|  |  |  |
| --- | --- | --- |
| **Objective** | **Importance** | **Justification** |
| 1. There needs to be a loading/welcome screen that greets the user  1.1 This displays the game’s logo and plays background music. | Preferred | It welcomes the user and gives them a first impression of what the game will be like. |
| 2. A main menu needs to be displayed with intractable buttons.  2.1That navigate the user to start the game, login and give instructions. | Essential | It allows the user to initialise and play the game as well as login to their separate account and learn how the game is played. |
| 3. The login button will give the user an option to login asking them to input their username and password. | Preferred | As the project is a game, a user account is not required as another person is not likely to be playing the game on the users’ computer. However, it will allow the user to access their own save files and see their statistics compared to other users on that local pc. |
| 4. The start game button will allow the user to initialise the game.  4.1They will be redirected to a character selection screen. | Essential | This lets the user initially start the game. Without this, they will only be able to access the main menu. |
| 5. The instructions button will redirect the user to a different screen  5.1This screen will inform them on how to play the game. | Essential | This gives the user the crucial information they will need to play the game, for example controls and rules. They will also be told the basic mechanics of the game for example, how items work and how the health system works. |
| 6. A character selection screen will be added displaying characters for the users to play as.  6.1 That can interact with to select. | Preferred | It gives the user the access to select the character they want to play as in that specific play through of the game and is required to start the game. |
| 7. The user will be able to select different characters. | Preferred | The different characters allow the user to experience different aspects of the game, as each character will be unique and have different attributes. This adds to the difficulty mechanic of the game and gives choice to the user. They are not required but helps the complexity of the game. |
| 8. The user can select an option in the login screen to create an account if they do not have one. | Essential | The user requires setting up their own account if they want to access and save their own game files as well as access the leader boards. |
| 9. An error message is displayed if the user has input their login information wrong. | Essential | Reduces errors in the game, stop users from accessing other account and informs the user that they have input their information wrong then asks them to input it again. |
| 10. The game will play background music as the user plays the game as well; as well as play sounds when the character is hit or collects an item. | Preferred | The music stops the user from getting bored as a silent game can have a dull atmosphere. Sounds also help indicate to the user when they are damaged or when they collect an item. |
| 11. The game has to have a movement system controlled by the user to move up, down, left and right. | Essential | The character needs to have a movement system so the user can progress through the game and avoid projectiles. This can be done by using WASD or Arrow keys. |
| 12. The player can control the character to strafe around the game space and a sluggish movement. | Preferred | This allows the user to strafe around the screen giving them more freedom to move around the level. The sluggish movement requires more complex programming and adds a difficulty so the player has to adapt to the movement. |
| 13. The when a player enters a new room the room will be generated. | Essential | This feature alters the players experience, as they will never experience a same play through, also it keeps the player more interested as they can discover different rooms as they play. This can be done by having pre-set rooms being generated. |
| 14. A type of room a player will enter is an item room. | Preferred | The player can only access items by entering one of these rooms, there will be only one of these rooms a game and they will randomly generate one item in the room. |
| 15. Another type of room a player will enter is an item shop. | Preferred | Items can be bought in these rooms by spending their coins what they collect by playing the game. Five items will be generated that have different prices. |
| 16. The game will spawn an AI controlled enemy when they enter a room. | Preferred | The game needs an enemy for the player to shoot and kill; this will be controlled by the computer as the game is intended to be a single player experience. |
| 17. The enemies generated will be different from each other and have different health and attack patterns. | Preferred | The different enemies’ gives variety to the game as well as increase the difficulty of the game as harder enemies will spawn later in the game. |
| 18. Items will spawn when the player enters a room. | Essential | Items give the player better abilities or penalise the player based on a random spawning system. In order for the players to get them, they need to be spawned in the game. |
| 19. Items will have different abilities and different buffs or penalising effects. | Preferred | The different effects of items effect the players character, due to this depending on which items they collect the game can become harder or much easier. The better items will be rarer to collect. |
| 20. Coins will spawn in the game for the player to collect. | Preferred | Coins will spawn when a player kills an enemy or they can be collected by them spawning in the room randomly as a gift. These coins will accumulate as the player collects them. |
| 21. The game will have a User interface showing the health, the items the character has, their money count and an interactive button to take you to the setting tab. | Essential | The user needs to be able to see their characters health, money and items as it helps them make decisions as they play the game; for example, they can view their money and then buy items form an item shop. They also need to access the settings tab to alter the game. |
| 22. The user interface will follow the same art style of the game as well as be small and transparent. | Essential | The user interface following the games art style gives a more comfortable feel to the user as it does not stand out too much and look abstract to the game. The user interface also has to be minimal and transparent so it does not obscure the user’s view and distract the user from the core game. |
| 23. The setting tab will show a separate screen that displays a music slider, sound slider and an intractable exit game button. | Preferred | Music will be playing in the background and use sounds so if the player gets distracted by the sounds they can turn them down using the slider provided. If the user wants to exit the game, they can do this by clicking the button that will redirect them to the main menu. |

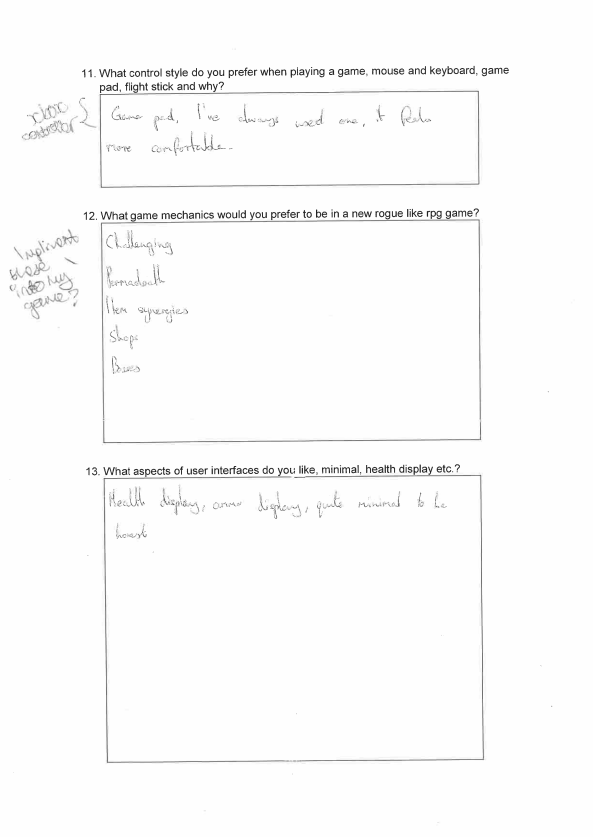
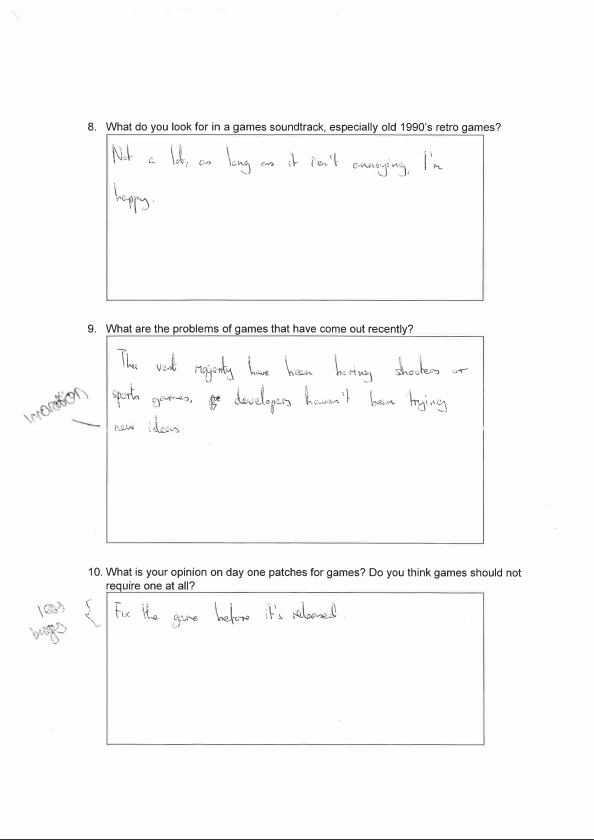
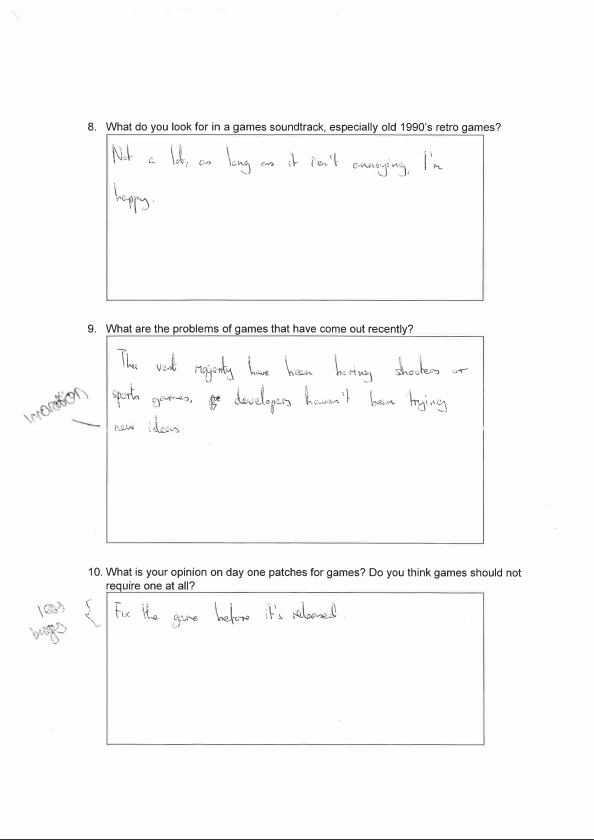
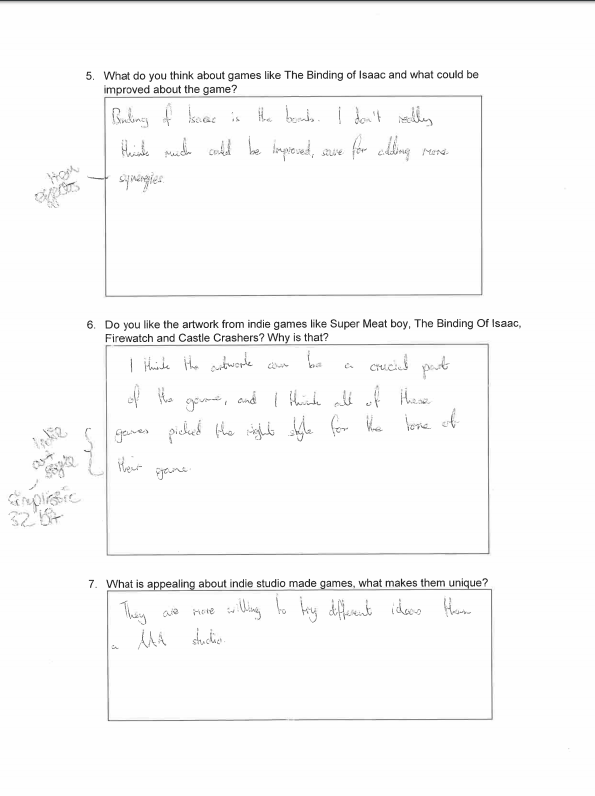
### Limitations

The game will be limited in many ways to shorten development time as well as reduce the complexity of the game. One of the limitations is that the game will not be online, the game will be strictly a single player experience, and the game does not need to be online as it is intended for only one player to play the game by themselves. The only online aspect of the game would be online is global leader boards that are not needed; these leader boards will be local to the system.

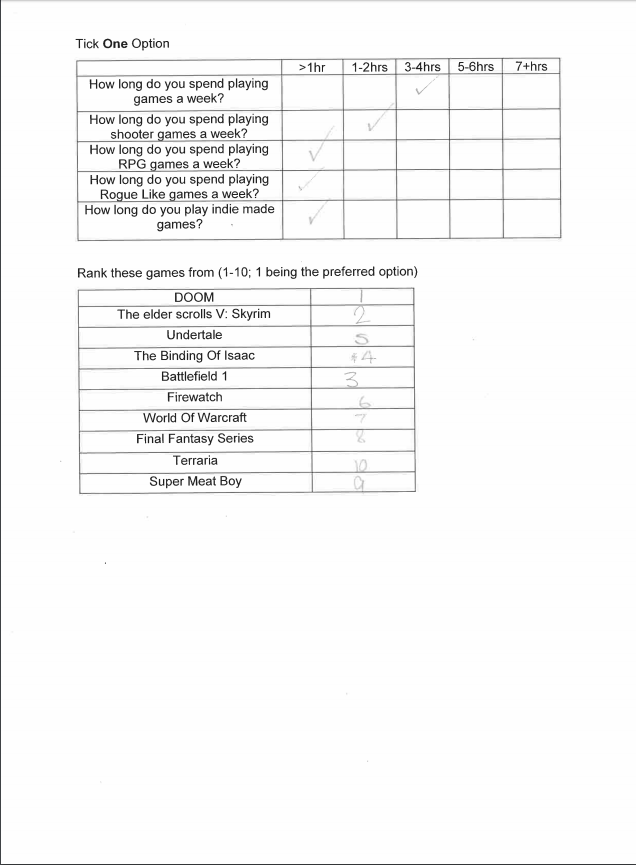
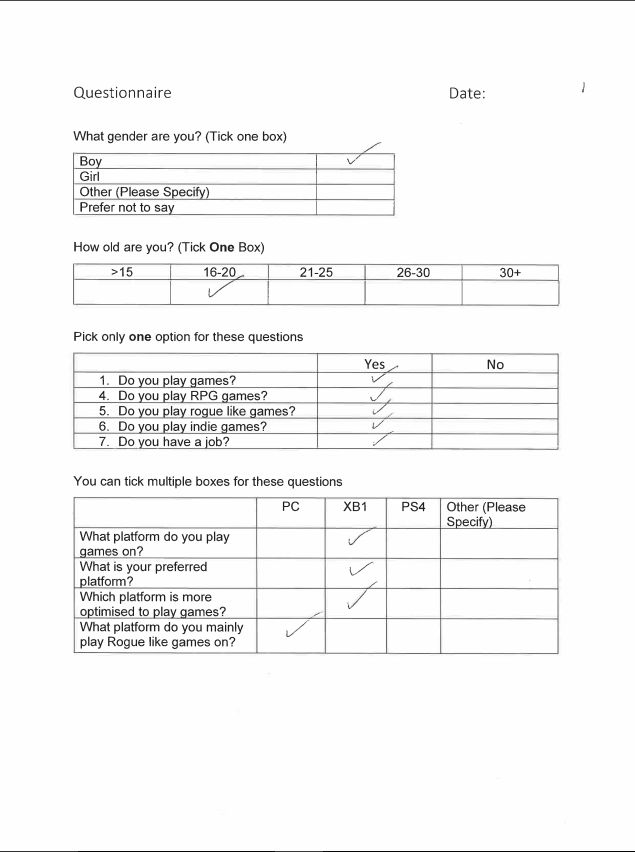
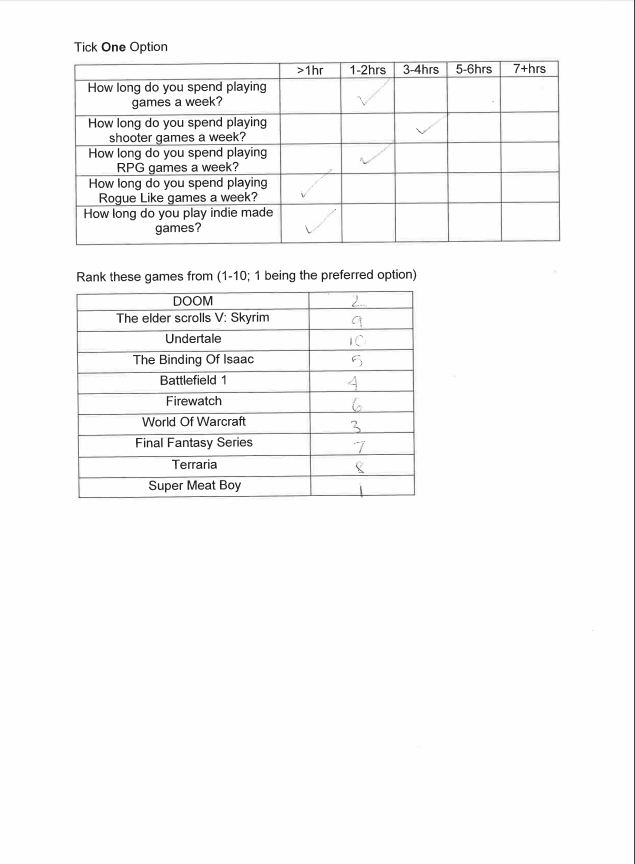
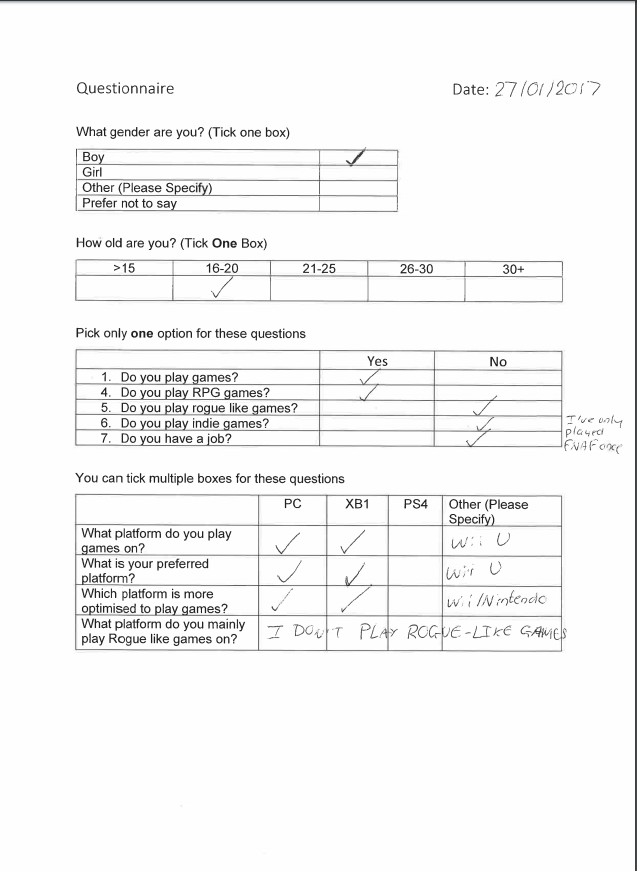
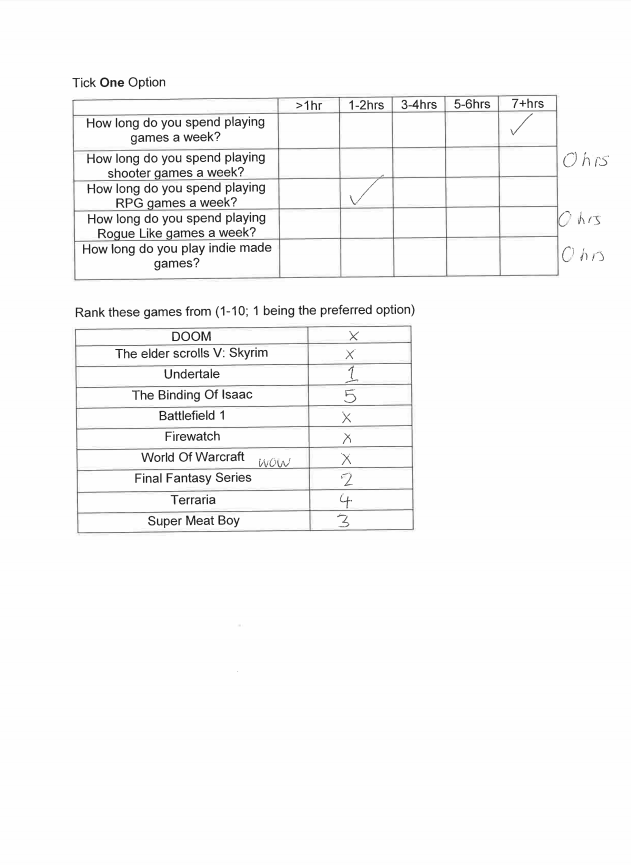
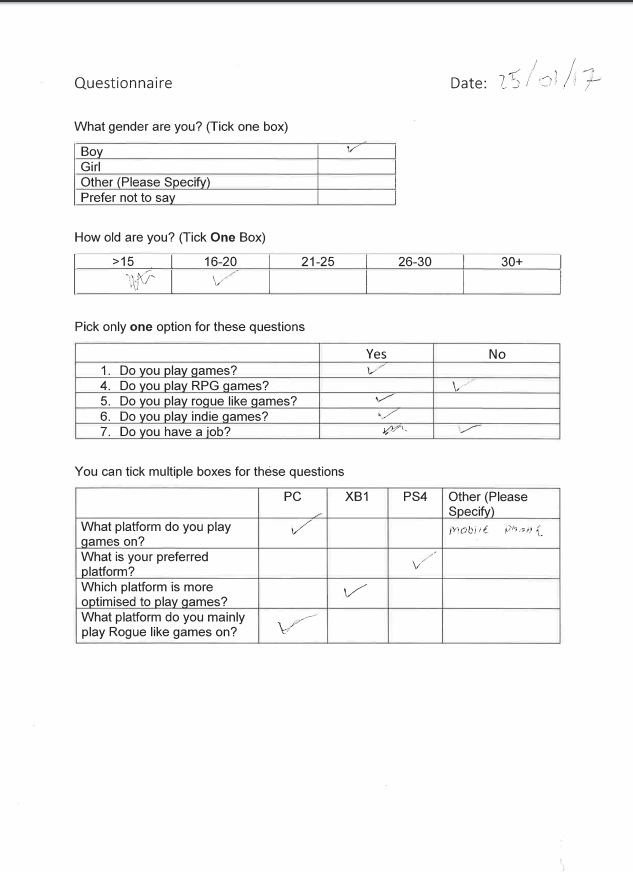
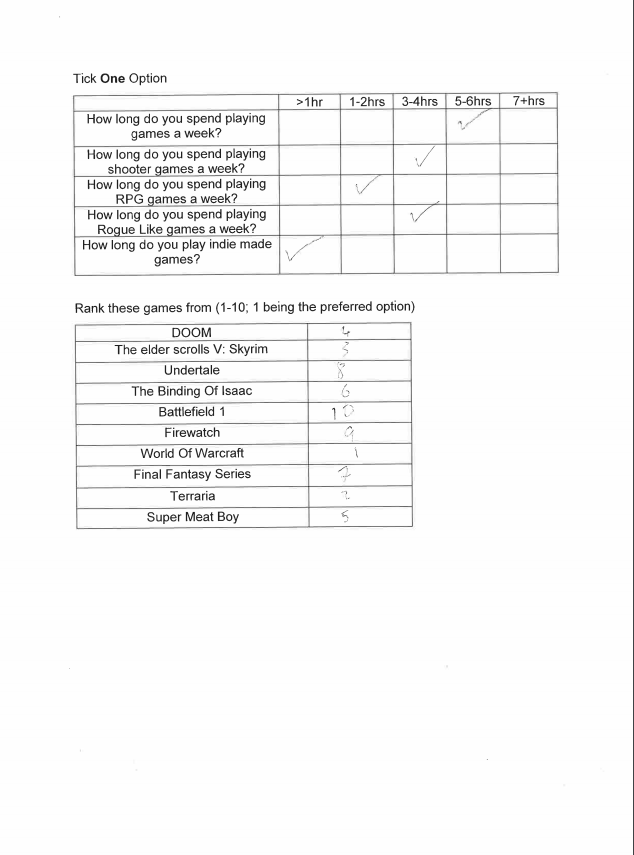
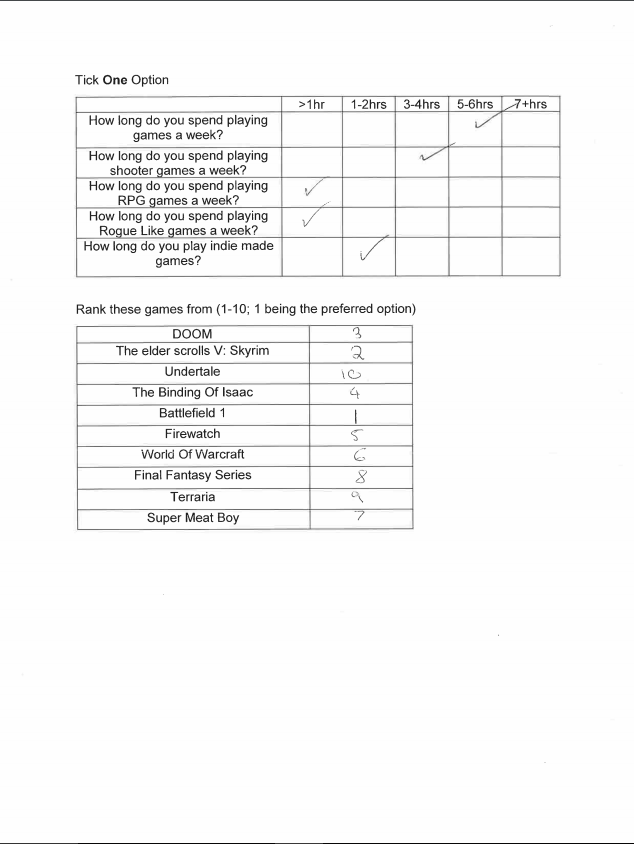
Another limitation is due to the game mainly being a single player experience there will be no two player/co-op experience. This means that another person will not be able to control a second character simultaneously with his or her friend or use two accounts at the same time.

The final limitation is that the game cannot be developed for other platforms than windows operating system, this is due to visual studio (the development software), only being compatible with Windows devices not android, apple operating system or mac operating system. For mobile devices, the game would have to be redesigned to fit the scale of phones, the layout of app games and the game would have to be more optimised to run on the phones.

## C:\Users\s013138\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Interview1.pngInterview Evidence



Questionnaire Evidence



# 2.Design

This design document will be used to help develop Volniir and design the program. It will structure out the program into different sections, this will help developers understand what the program will include and what different sections are built up of.

**Volniir**

## 2.1.1 Modular Design

Enemies

Player

Enemy

Tears

Items

Static Entities

Rocks

Dynamic Entities

Rooms

Object orientated levels with different appearances, objects and items

Waves

Object orientated designed waves that vary and change some entities attributes. When max wave is reached the game exits

Character 1&2

Depending on which character the player chose their movement speed, appearance and projectile attributes will change

Instructional Screen

Teaches the user how to play the game. Controls and info

Leader boards

A leader board is displayed; this shows different user’s progress

Cutscene

Gives a lore explanation to the player

Main Menu

The menu helps the user navigate the program so they can start the game, exit the game and view a cut scene. The Instructions are displayed on the screen to help the user

Game Screen

This screen is the main game, it allows the user to play the game and experience the features, lore and gameplay

Character selection screen

This screen gives the user to select a character and play the game

Exit Game

The game fully exits and saves statistics.

### 2.1.2 Modular Design Explanation

This hierarchy has been laid out this way because it helps developers follow how the game is structured. The layout breaks the game from the introduction screen to the game screen. There are different types of logins to choose depending on what user is using the game. Each screen has multiple options to give users choice and change their gameplay experience. Developers can then program all these features in detail as the chart explains what each section includes. The modular design allows each screen to be tested separately in detail.

### Class association

My modular design compliments class association as different classes can use Dependency to use other class’ attributes, for example using the position of the player for a projectile to spawn at.

Aggregation is a class association that allows a “player” to buy an attribute from another class and it not depend on that class. So I could use aggregation to spawn more items or rocks during gameplay.

### Class interaction

Object orientated programming is going to be a large aspect of the game. Most variables and attributes will be stored or accessed through the Game screen as most entities such as the player or the enemies will need to use other class attributes or variables for example the enemy loses 10hp as the player’s projectiles do 10 damage. As well, the waved based system I have designed relies on the ability to be able to spawn different environments, enemies or rocks while the game screen is running; having most attributes and variables stored in the game screen will allow me to create new instances.

Game Screen

The lines represent the classes and attributes that can be called by, changed and stored in the Gamescreen class.

## 2.2 The Solution

### 2.2.1 Data Structure

**Game Data**

**User**

### Game Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Data Type | Validation | Justification | Key Type |
| DataID | Integer | Presence Check  Valid Integer | Unique identifier for the data | Primary |
| UserID | Integer | Presence Check  Valid Integer | Links a user to the game data | Foreign |
| Health | Whole Number | Presence Check  Valid Whole Number  Valid Range | Stores the user’s character health (0-10) |  |
| ItemList | String | Presence Check  Valid String | Stores the items the user currently has |  |
| Money | Integer | Presence Check  Valid Integer | Stores the amount of money the user has |  |
| Life | Boolean | Presence Check (yes or no) | Determines if the user’s character is alive or not |  |
| GameTime | Whole Number | Presence Check  Valid Whole Number | Stores how long the user has been playing the game. |  |

## User Interface Designs

I have used adobe photoshop to design my screens, as it is easy to create shapes, line the shapes accurately and then easily scale the shapes. Photoshop also has a wide variety of text fonts to use.

### Introduction Screen



Usability

When the user initially launches the game, they are visited with the logo of the game including an option to start the game.

Validation

This screen is just to greet the user and give them an idea on how the game will look. There is only one option, which is a button that can be clicked; therefore, there will not be any validation.

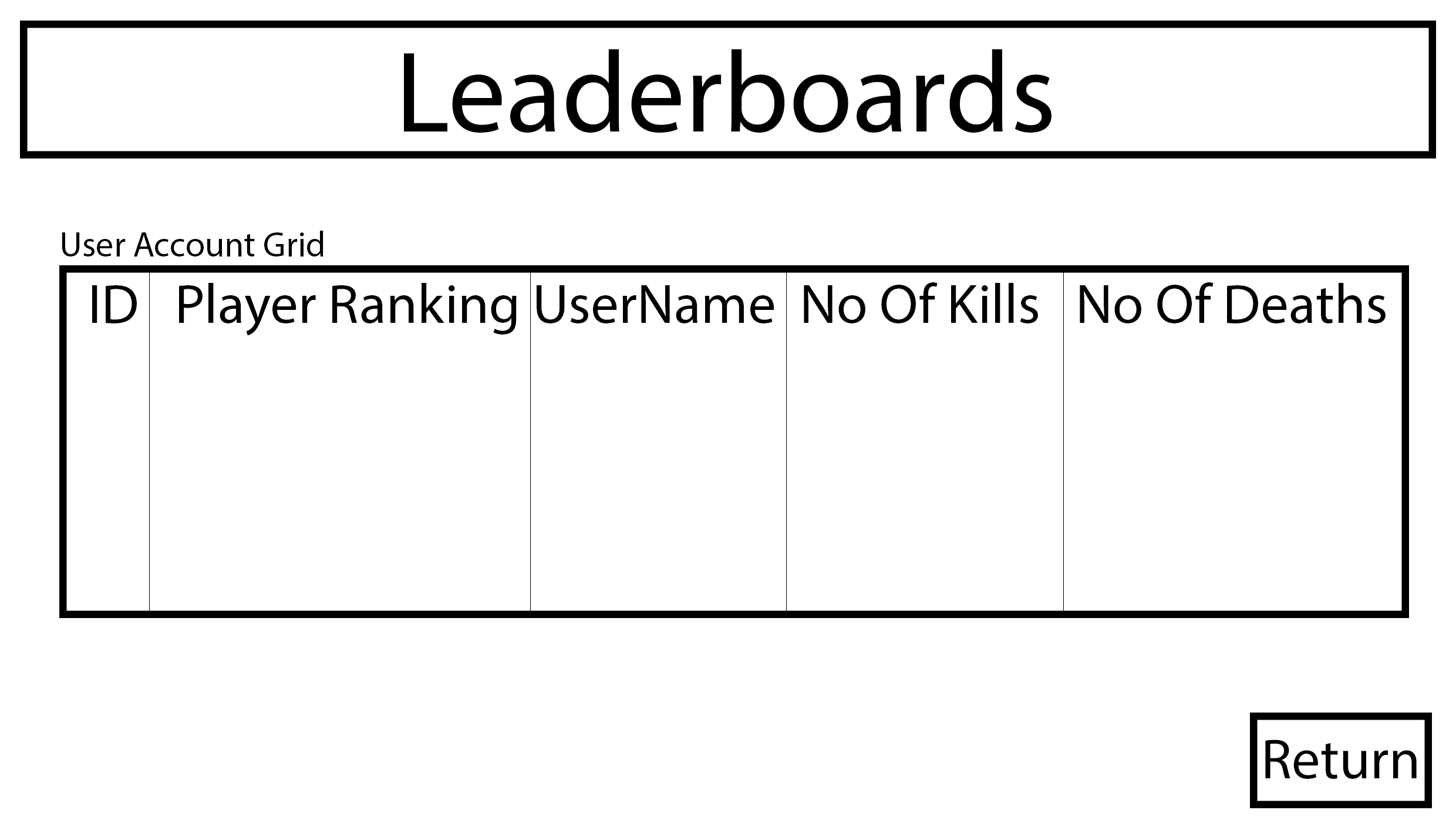
Algorithms

Start Game Button

OPEN LoginScreen

CLOSE IntroScreen

### Leader boards



Usability

If the user selects the leader boards they are displayed all of the users ranked by the number of kills they have while they have been progressing through the game. However, the leader board may also display the user’s personal statistics by themselves so they can perfect each run of the game.

Validation

There is no validation for this screen as the user can only use buttons to go to another screen.

Algorithms

USER ACCOUNT GRID(Imports the data from the leader record file and displays that data in the table)

WHILE (NotEndOfFile.LeaderRecord)

READ LineFromFile

IF (LineFromFile.Active == TRUE)

LeaderboardTableRows = LineFromFile.Split ”,”

END IF

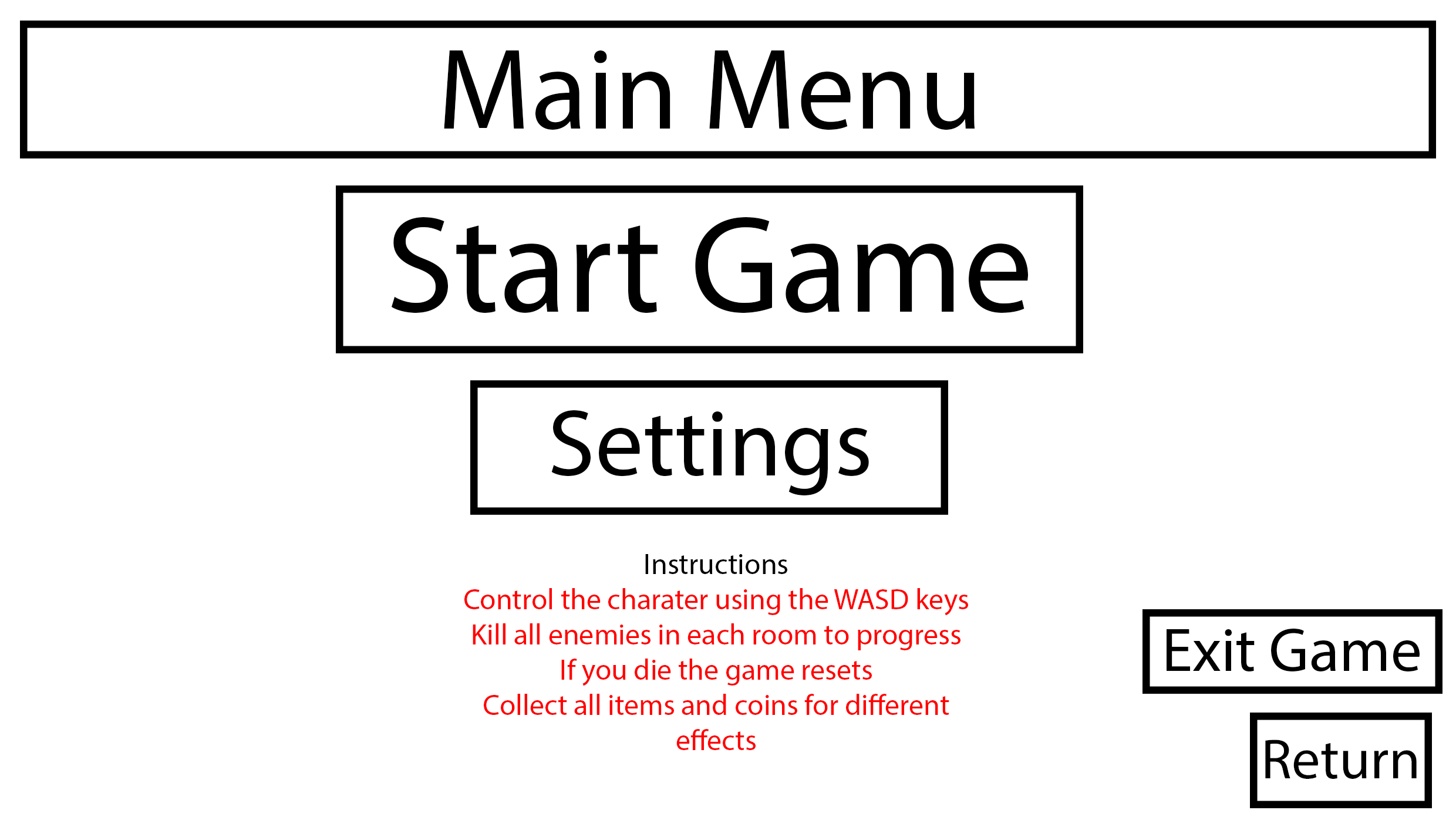
END WHILE

RETURN BUTTON(Used to return to the previous screen)

OPEN UserScreen

CLOSE Leaderboards

### Main Menu



Usability

When the user, launches the game they are greeted with this screen that gives the user the options to go to accesses the game’s settings, exit the game, return to the login screen and start the game. The game’s instructions are also displayed on the screen informing the user.

Validation

Due to this screen only containing buttons there will be no validation needed.

Algorithms

START GAME BUTTON

OPEN StartScreen

CLOSE Mainmenu

SETTINGS BUTTON

OPEN SettingsScreen

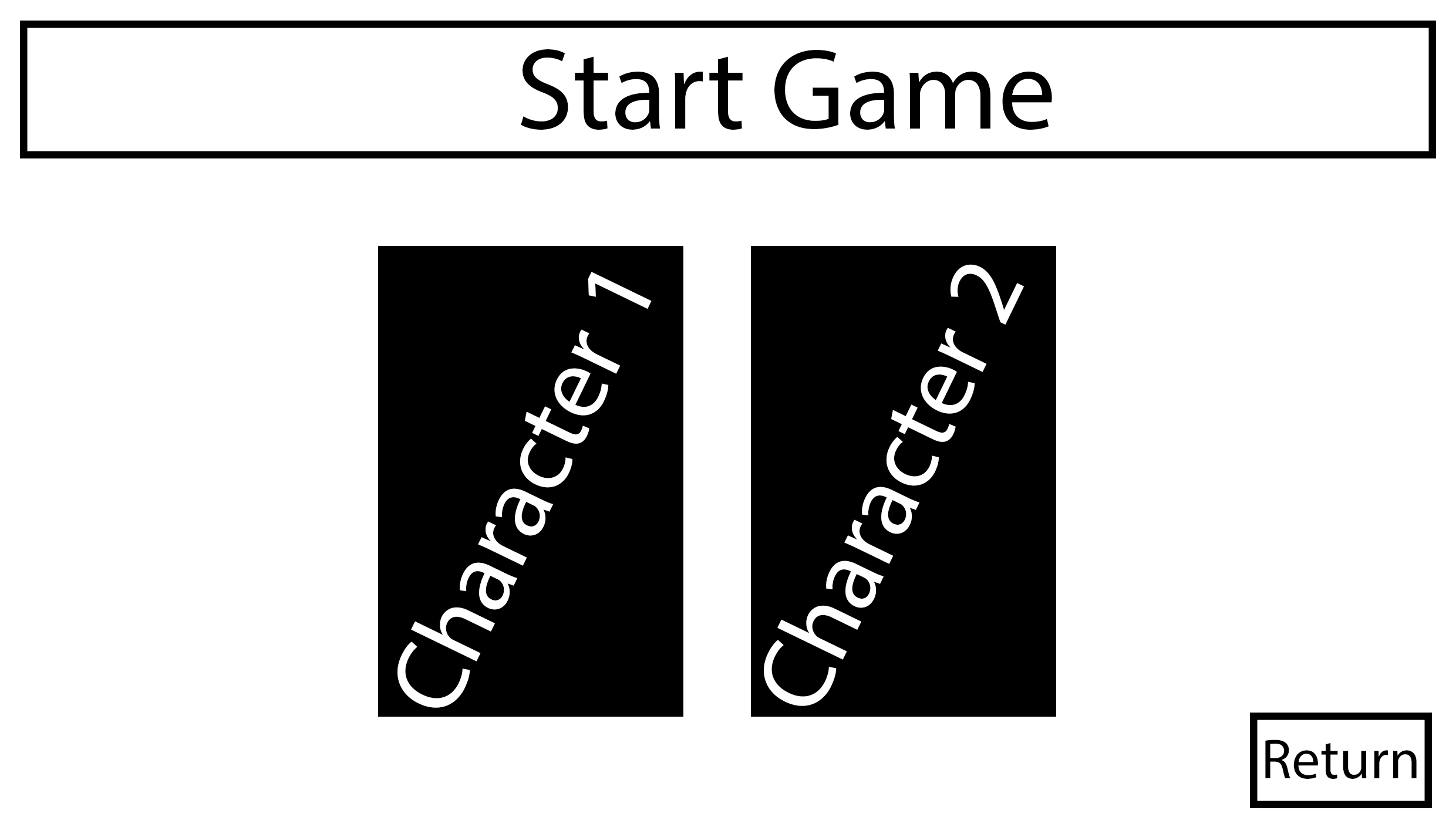
CLOSE MainMenu

RETURN BUTTON(Used to return to the previous screen)

OPEN MainMenu

CLOSE SettingsScreen

### Start Game



Usability

When the user wants to start the game, they are greeted with this screen where they can select what character they want to play as, this allows the player to experience the game in two different ways creating further playability.

Validation

There will be no validation needed as the user can only use buttons on the screen.

Algorithms

CHARACTER 1 BUTTON(Sets character 1 as the user’s character)

CHARACTER = “CHARACTER 1”

CHARACTER 2 BUTTON(Sets character 2 as the user’s character)

CHARACTER = “CHARACTER 2”

RETURN BUTTON(Used to return to the previous screen)

OPEN GameScreen

CLOSE StartScreen

### Exit Game



Usability

When the user wants to exit the game, they are presented with this screen to verify if they want to exit the game.

Validation

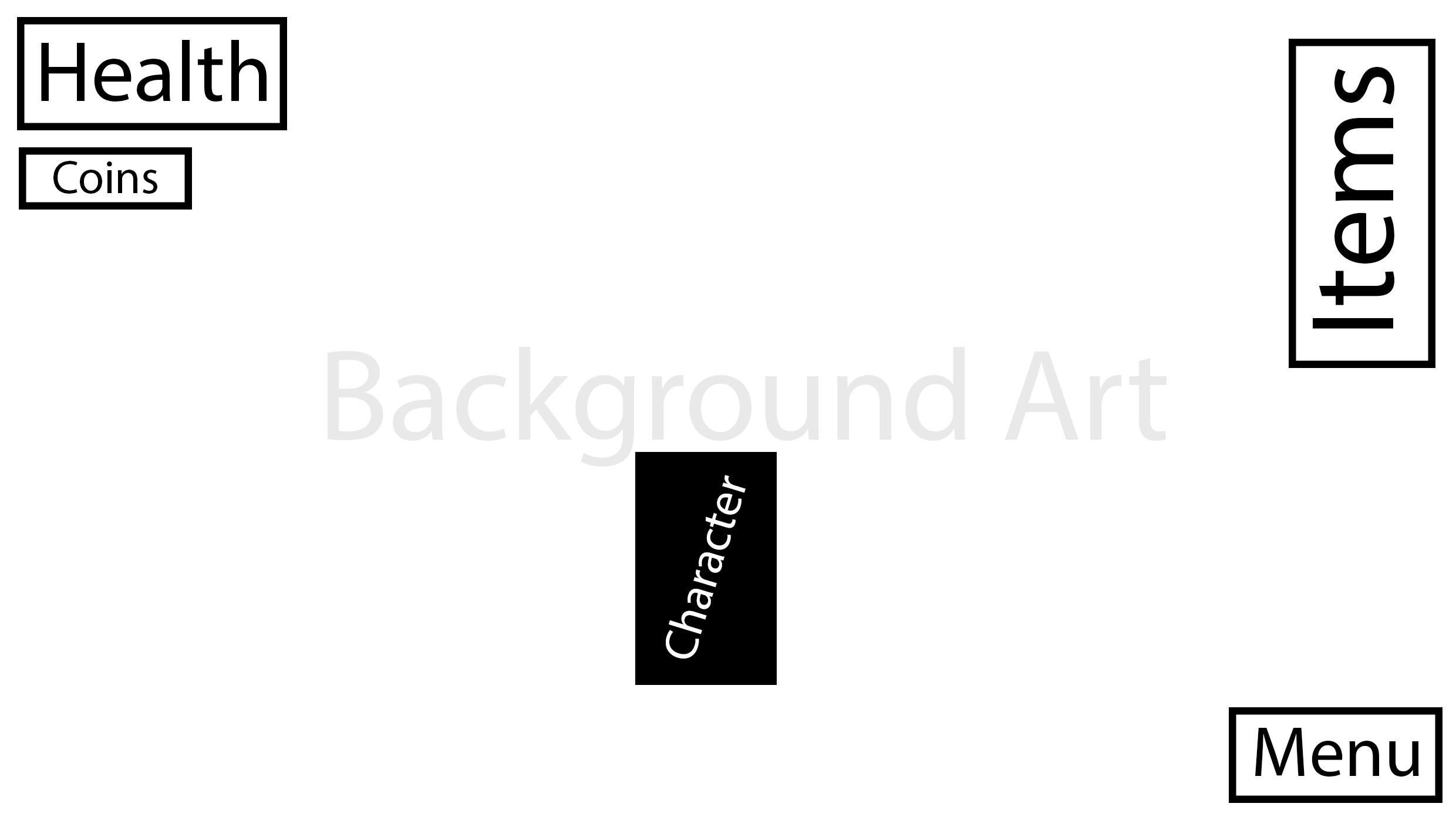
There will be no validation needed as the user can only use buttons on the screen.

Algorithms

EXIT GAME BUTTON(Used to exit the program)

EXIT program

### Game Screen



Usability

This is the main screen for the game where most of the game is played on. The character is displayed on screen as well as their health, coin count and items. The user has the option to return to the main menu whenever they want.

Validation

There will be no validation needed as the user can only use buttons on the screen.

Algorithms

MENU BUTTON(Directs the user to the menu screen)

OPEN MainMenu

CLOSE GameScreen

HEALTH COUNTER (Displays the users health)

HEALTH = GLOBALHEALTH

OUTPUT HEALTH

COINS COUNTER (Displays the users total coins)

COINS = GLOBALCOINS

OUTPUT COINS

CHARACTER(Controls the character’s direction)

x = pictureBoxLocationX

y = pictureBoxLocationY

IF(KeyCode == KeysRight)

x += 2

ELSEIF(KeyCode == KeysLeft)

x -= 2

ELSEIF(KeyCode == KeysUp)

y -= 2

ELSEIF(KeyCode == KeysDown)

ENDIF

IF(pictureBox2.Image == MoveRight)

y += 2;

ELSEIF(pictureBox2.Image == MovieLeft)

y += 2;

ENDIF

## Testing Strategy

### Testing Strategy Description

Testing is a key feature in the development of the project, this section of development helps to see if each modular section of the project works. Development and testing logs will be used to document and organise each test that will be carried out; these logs will be split into three different tables: During development, User testing, After development testing.

White box testing will be used to test each modular feature while the code is accessible to the developer, this allows the developer to quickly alter the code after each test.

Black Box Testing will be used by the developer to test the program and only view the inputs and outputs of the program, this will be used after white box testing is complete.

### System Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No. | Test | Test Data | Type Of Data | Test Justification | Expected Result |
| 1 | Test to see if different characters can be used. | Test: Character1 | Normal | To check to see if the multiple character’s can be used. | When the game is ran only character 1 will be loaded in and playable. |
| 2 | Test to see if character movement works. | Keyboard: WASD | Normal | To check if the movement system works | When W is pressed the character moves up, S the character moves down, A character moves left, D character moves right. |
| 3 | Test to see if the character can strafe. | Keyboard: WASD | Normal | To check if the character can strafe | When W+D or W+A are pressed the character moves diagonally. |
| 4 | Test to see if the walking animation works. | Keyboard: WD | Normal | To check if the animation is triggered. | When a movement key is pressed the animation is triggered, |
| 5 | Test to see if an item is spawned | Item room  Item: Coin | Normal | To check if an item can be picked up and detected | When entering the item room a coin will be displayed on screen. |
| 6 | Test to see if an item changes character attributes. | Item room  Item: Sword | Normal | To check to see if items can affect attributes. | When the sword is picked up the character’s damage will be changed. |
| 7 | Test to see if the character can collide with walls | Keyboard: WASD  Default Room | Normal | To check if the character collides with solid objects | The character will stop when hitting a wall. |
| 8 | Test to see if the start game button works | Start Game Button  Intro Screen | Normal | To check if the button directs the user to the login screen. | When the button is pressed the screen will change to login. |
| 9 | Test to see if the timer works | Time = 0  Game Screen | Normal | To check if the times increases as the player uses the game | When the player is on the game screen the timer should increase every second and display hours, minutes and seconds |

# Development

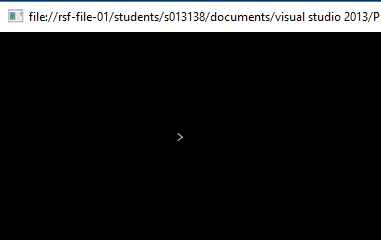
Prototypes will be made to test one of the fundamental features of the game which is the movement of a character. Each prototype will be made in different ways to see which prototype is the best, and what the final solution should be made on.

## Prototype 1

Prototype 1 demonstrates a simple way of moving a character across a screen in both x and y directions using the co-ordinates in a console application via C#.

This prototype uses key detection and a co-ordinates variable to set the position of the cursor and write the character.

Code source: https://stackoverflow.com/questions/33924505/c-sharp-console-application-moving-a-character



### Character movement code

case ConsoleKey.DownArrow: // When the down arrow pressed y-coordinate + 1

y++;

break;

case ConsoleKey.UpArrow: //Up Arrow arrow pressed y-coordinate - 1

if (y > 0) //Only executes when the character's y position is bigger than 0 to reduce errors

{

y--;

}

break;

case ConsoleKey.LeftArrow: //When the left arrow pressed x-coordinate - 1

if (x > 0)//Only executes when the character's x position is bigger than 0 to reduce errors

{

x--;

}

break;

case ConsoleKey.RightArrow: //When the Right arrow pressed x-coordinate + 1

x++;

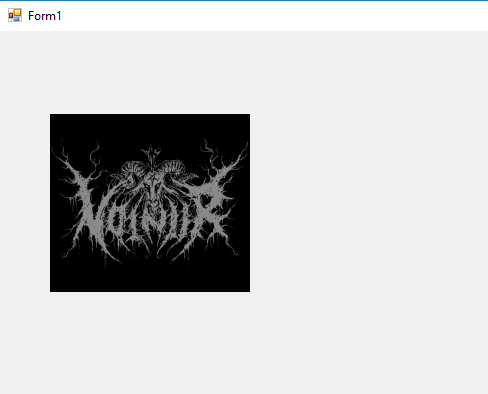
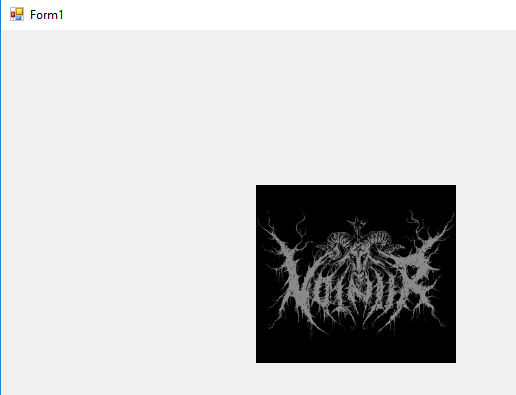
break;

}

This code can be translated into my final piece as a method of moving the game character across the screen. However as one of my objectives is to implement a strafing movement style I will have to design the code so both co-ordinates can be changed simultaneously.

## Prototype 2

For prototype 2 I wanted to implement the idea of using co-ordinates to move a character but develop it in a windows application using a picture image as the character being used.



### Character movement code

int x = pictureBox2.Location.X;

int y = pictureBox2.Location.Y;

pictureBox2.Visible = true;

{

if (e.KeyCode == Keys.Right)

{x += 6;}

else if (e.KeyCode == Keys.Left)

{ x -= 6;}

else if (e.KeyCode == Keys.Up)

{ y -= 6;}

else if (e.KeyCode == Keys.Down)

{ y += 6;}

else if(e.KeyCode == Keys.Down & e.KeyCode == Keys.Right)

{

y += 6;

x += 6;

}

else if(e.KeyCode == Keys.Down & e.KeyCode == Keys.Left)

{

y += 25;

x -= 25;

}

else if(e.KeyCode == Keys.Up & e.KeyCode == Keys.Right)

{

y -= 25;

x += 25;

}

else if(e.KeyCode == Keys.Up & e.KeyCode == Keys.Left)

{

y -= 25;

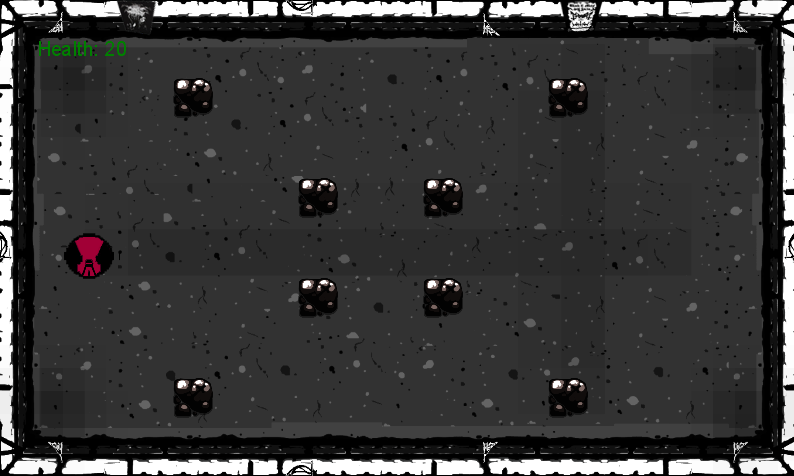
x -= 25;

}

}

## Prototype 3

For prototype 3 I wanted to start development of the game using the visual studio extension monogame. Monogame allows me to program my own velocities and complex movement systems using complex mathematics.



Character movement code

// Use the Keyboard

if (currentKeyboardState.IsKeyDown(Keys.Left))

{

player.Position.X -= playerMoveSpeed;

}

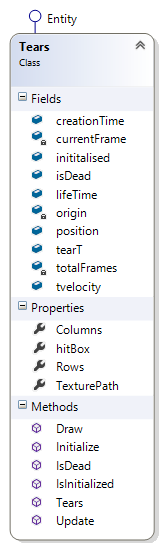
if (currentKeyboardState.IsKeyDown(Keys.Right))

{

player.Position.X += playerMoveSpeed;

}

# Technical Documentation



## Class Association

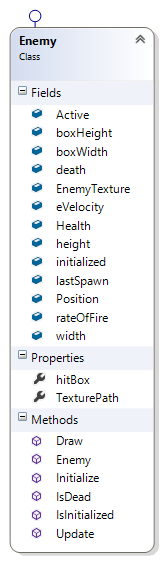
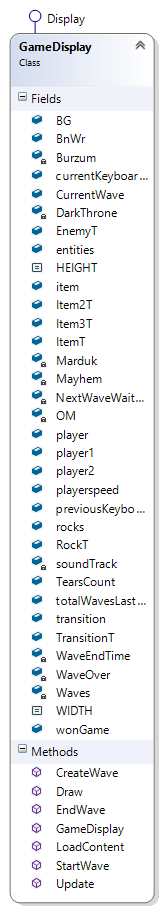
## Dependency

Wants to look at all the rocks in…

### Tears & GameDisplay

|  |  |
| --- | --- |
| foreach (Rock r in Game1.GAME.GameDisplay.rocks)  {  //If the rock's health is less than or equal to zero the rock is concidered dead  if (r.health <= 0)  {  r.isDead = true;  }  if (hitBox.Intersects(r.hitBoxR))  {  //If the tear hits the rock then the velocity is reversed  tvelocity \*= -1;  if (Game1.GAME.GameDisplay.player.hasItem)  {  r.health -= 50;  isDead = true;  }  } | Code explanation  The tears class assigns the list of rocks stored in the GameDisaplay so it can then look through that list to see if a tear intersects with the certain rock it is intersecting with.  There is also a condition where if the player has picked up an item and sets the player’s attribute “hasItem” to true then the player’s tears can do damage to the rocks. |

### Enemy & GameDisplay & Game1



//Checks enemy to see if they interset with any tears

foreach (Entity e in Game1.GAME.GameDisplay.entities)

{

if (e is Tears) {

if (hitBox.Intersects((e as Tears).hitBox))

{

Game1.GAME.GameDisplay.player.TLanded++;

Health -= Game1.GAME.GameDisplay.player.playerDamage;

(e as Tears).isDead = true;

}

}

}

//Checks to see when the last time an enemy tear has spawned added wtih the rate of fire is less

//than the total game time elapsed in seconds

if (lastSpawn + rateOfFire < gameTime.TotalGameTime.TotalSeconds)

{

//If so create a new tear into a list & update the time a tear has spawned

ETears eT = new ETears(gameTime.TotalGameTime.TotalSeconds, new Vector2(this.Position.X, this.Position.Y), 2, 2);

lastSpawn = gameTime.TotalGameTime.TotalSeconds;

if (!eT.inititalised)

{

eT.Initialize(Game1.GAME.Content.Load<Texture2D>(eT.TexturePath));

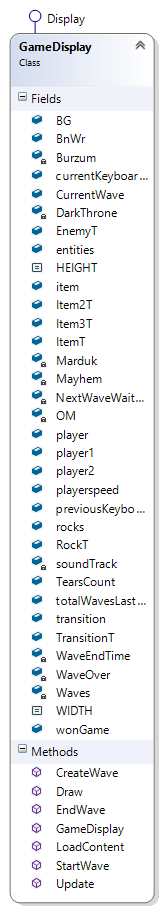
}

Game1.GAME.GameDisplay.entities.Add(eT);

The enemy class searches through each enemy in the entities list stored in gamedisplay, this is so it can see if a player tear has intersected with the enemy’s hitbox, if so the tears landed attribute increments by 1 and the enemies health decreases by the player damage which is also stored in gamedisplay.

The enemies’ class also uses the Game1 attribute gameTime that allows the enemy class to store how long the game has been running for, this is passed into and stored in the enemy class. The enemy’s class uses this to automatically spawn the enemy’s tears at set intervals. When a new tear can spawn it is initialised using Content.Load in Game1 and then adds an enemy tear into the entities list stored in gamedisplay.

### CharacterSelectionDisplay & GameDisplay

//Checks to see if the mosue is within the button boundaries, if so and the mouse is clicked player 1 is selected

//For the Player 1 button

if (state.Position.X > 310 && state.Position.X < 310 + boxWidth

&& state.Position.Y > 350 && state.Position.Y < 350 + boxHeight)

{

if (state.LeftButton == ButtonState.Pressed)

{

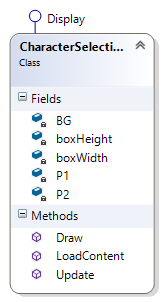
Game1.GAME.GameDisplay.player1 = true;

Game1.GAME.GameDisplay.player.PlayerTexture = Game1.GAME.Content.Load<Texture2D>("Assets\\p2SS");

Game1.GAME.GameDisplay.player.playerMoveSpeed = 2;

Game1.GAME.GameDisplay.TearsCount = 1;

Game1.GAME.Leaderboard.refresh();

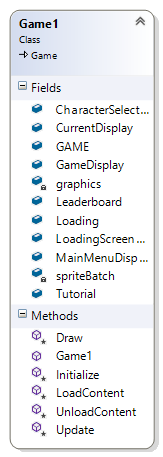
 Game1.GAME.CurrentDisplay = Game1.GAME.GameDisplay; }

}

In characterselectiondisplay if player1 is selected it sets the values of player1 attributes that are stored in the gamedisplay including passing through the texture for the character to gamedisplay.

As well it sets the currentdisplay variable to the gamedisplay, so the screen switches to the game.

### MainMenuDisplay & Game1



//Checks to see if the mosue is within the button boundaries, if so and the mouse is clicked change screens

//For the exit button

if (state.Position.X > 310 && state.Position.X < 310 + boxWidth

&& state.Position.Y > 350 && state.Position.Y < 350 + boxHeight)

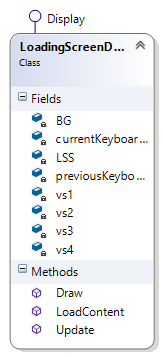
{

if (state.LeftButton == ButtonState.Pressed)

{ Game1.GAME.Exit(); }

In this code, the MainMenuDisplay checks the mouse position to see if it is within the boundaries of a button and has clicked, if so it exits the Game.

### LoadingScreenDisplay & Game1



else if (gameTime.TotalGameTime.TotalSeconds >= 55 || vs4)

{

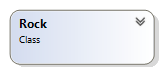
Game1.GAME.CurrentDisplay = Game1.GAME.CharacterSelectionDisplay;

BG = Game1.GAME.Content.Load<Texture2D>("Assets\\LoadingScreen\\VLS5"); vs4 = true;

}

In this subroutine if the total time elapsed while the LoadingScreenDisplay is running is larger than 55 seconds the Game1 current display variable is changed to the CharacterDisplay that makes the game change screens. As well, the background texture is passed through the content load function, accessed in Game1, so the background for the LoadingScreenDisplay is “VLS5”.

## Aggregation



Buys 5 rocks

### Rocks & GameDisplay

|  |  |
| --- | --- |
| Algorithm  //Represents the rock and creates a list of rocks to be stored in  public List<Rock> rocks = new List<Rock>();  //Sets the X&Y-Co-ordinate within the game viewport at a proportional gradient  Rock r = new Rock(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - i \* 100, RockT);  Rock r2 = new Rock(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - (i \* -100 + 500), RockT);  rocks.Add(r);  rocks.Add(r2); | Explanation  When this subroutine is called the GameDisplay class “buys” rocks from the Rock class and adds them to the rock list stored in Game1. The two different types of rock (r & r2) are used to create symmetrical patters of rocks for each “room”. |

### Items & GameDisplay

|  |  |
| --- | --- |
| Algorithm  //Represents the item spawned  public List<Item> item = new List<Item>();  //Adds 1 item into the item list  Item iT = new Item(Game1.GAME.GraphicsDevice.Viewport.Width / 2, Game1.GAME.GraphicsDevice.Viewport.Height / 2);  iT.Initialize(ItemT);  item.Add(iT); | Explanation  When the subroutine is called the GameDisplay class “buys” an item and stores it in the item list stored in GameDisplay. It also initialises the item so it can be drawn into the game. |

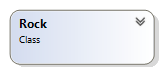
### Entities & GameDisplay

|  |  |
| --- | --- |
| Algorithm  //Represents the enemies, tears and enemy tears  public List<Entity> entities = new List<Entity>();  //Adds 5 enemies into the enemy list  for (int i = 0; i < 5; i++)  {  //Sets the X&Y-Co-ordinate within the game viewport at a proportional gradient  Enemy e = new Enemy(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - 60, EnemyT);  entities.Add(e);  } | Explanation  The entities list is a combined list that contains any class associated with the entities interface (Enemies, tears, enemy tears). The game display can spawn 5 enemies and store them in the entities list. |

### Tears & GameDisplay

|  |
| --- |
| Algorithm  ////Adds a tear into the list tear and send the corresponding key press that has been detected  if (previousKeyboardState.IsKeyDown(Keys.W) && currentKeyboardState.IsKeyUp(Keys.W) && entities.Count(entity => entity is Tears) <= TearsCount)  {  Tears t = new Tears(Keys.W, creationTime, 4, 4, tearPosition);  entities.Add(t);  } |

## Composition



1 Game display owns 1

### Rocks & GameDisplay

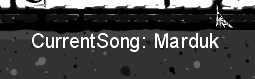
|  |  |
| --- | --- |
| Algorithm  //Sets the X&Y-Co-ordinate within the game viewport at a proportional gradient  Rock r = new Rock(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - i \* 100, RockT);  Rock r2 = new Rock(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - (i \* -100 + 500), RockT); | Explanation  Here is an instance where the rocks are “spawned” in by using the rock class to create a rock as well as passing through the rocks texture. The co-ordinates are mathematically calculated by getting the boundaries of the whole screen and multiplying by 125 for the x- co-ordinate or (i\*-100+500) for the Y- co-ordinate. |

### Enemies & GameDisplay

|  |  |
| --- | --- |
| Algorithm  //Adds 5 enemies into the enemy list  for (int i = 0; i < 5; i++)  {  //Sets the X&Y-Co-ordinate within the game viewport at a proportional gradient  Enemy e = new Enemy(Game1.GAME.GraphicsDevice.Viewport.Width - (i + 1) \* 125, Game1.GAME.GraphicsDevice.Viewport.Height - i \* 60, EnemyT);  entities.Add(e);  } | Explanation  Enemies are spawned into the Game Display where their co-ordinates are mathematically calculated and passed through to the Enemy Class as well. |

## Soundtrack

The soundtrack for the game was a challenge to program into the game as mathematics was required to create a song queue which isn’t implemented in monogame. I had to produce a class where the songs were stored and passed from the GameDisplay into the Soundtrack class.



### Algorithm

//List to store the songs

List<Song> queue = new List<Song>();

Song CurrentSong;

int CurrentSongIndex = 0;

TimeSpan CurrentSongStart;

I had to manually make a song queue so I could store and execute my songs one by one and then detect when the queue has finished, then loop back to the first song. This could be considered as a circular queue as the queue repeats itself and can be expanded.

public SoundTrack(params Song[] Songs) {

queue.AddRange(Songs);

}

public void Update(GameTime gameTime)

{

//If there is no current song playing, play the first song in the queue

if (CurrentSong == null)

{

CurrentSongIndex = 0;

CurrentSong = queue[CurrentSongIndex];

MediaPlayer.Play(CurrentSong);

CurrentSongStart = TimeSpan.FromSeconds(gameTime.TotalGameTime.TotalSeconds);

}

else

{

if (gameTime.TotalGameTime.TotalSeconds > CurrentSongStart.TotalSeconds + CurrentSong.Duration.TotalSeconds)

{

//If the last song has played restart the queue of songs

if (CurrentSongIndex == queue.Count)

{

CurrentSongIndex = 0;

}

//Once a song has finished increment the currentsongindex and play the next song

CurrentSongIndex++;

CurrentSong = queue[CurrentSongIndex];

MediaPlayer.Play(CurrentSong);

CurrentSongStart = TimeSpan.FromSeconds(gameTime.TotalGameTime.TotalSeconds);

}

}

}

## Leaderboards Screen



The leader boards screen displays the players local statistics from launching and playing the game the first time, it updates every time the user starts and exists the game wither by dying or pressing one of the exit game buttons on various screens.

The leader boards screen uses binary reader to read the generated file “LeaderBoards”

### Binary Reader

public void refresh()

{

//Reads the file and assigns the stored values to local variables

BnRd = new BinaryReader(File.Open("LeaderBoards", FileMode.Open));

wavesLasted = BnRd.ReadInt32(); <- Retrieves and stores the waves lasted integer value

tearsLanded = BnRd.ReadInt32(); <- Retrieves and stores the next integer value

enemiesKilled = BnRd.ReadInt32(); <Retrieves and stores the third integer value

itemsPicked = BnRd.ReadInt32(); <- Retrieves and stores the fourth integer value

wonGame = BnRd.ReadBoolean(); <- Retrieves and stores the Boolean value Won Game

BnRd.Close();

//Reads the file and assigns the stored values to local variables as a new copy

BnRd = new BinaryReader(File.Open("LeaderBoards", FileMode.Open));

wavesLasted2 = BnRd.ReadInt32();

tearsLanded2 = BnRd.ReadInt32();

enemiesKilled2 = BnRd.ReadInt32();

itemsPicked2 = BnRd.ReadInt32();

wonGame2 = BnRd.ReadBoolean();

BnRd.Close();

}

public void Draw(GameTime gameTime, SpriteBatch spriteBatch)

{

//Displays the statistics on screen

spriteBatch.Begin();

spriteBatch.Draw(BG, new Rectangle(0, 0, 800, 480), Color.White);

spriteBatch.DrawString(Game1.GAME.GameDisplay.player.HP, "Waves Lasted: " + wavesLasted2, new Vector2(300, 150), Color.White);

spriteBatch.DrawString(Game1.GAME.GameDisplay.player.HP, "Tears Landed: " + tearsLanded2, new Vector2(300, 175), Color.White);

spriteBatch.DrawString(Game1.GAME.GameDisplay.player.HP, "Enemies Killed: " + enemiesKilled2, new Vector2(300, 200), Color.White);

spriteBatch.DrawString(Game1.GAME.GameDisplay.player.HP, "Items Picked Up: " + itemsPicked2, new Vector2(300, 225), Color.White);

spriteBatch.DrawString(Game1.GAME.GameDisplay.player.HP, "Won the game: " + wonGame2, new Vector2(300, 250), Color.White);

To display the statistics on the screen I pass through a spritefont into the Leaderboards class and I use the Drawstring function to draw the test on screen.

spriteBatch.End();

}

### Binary Writer

Binary writer is used to initially create a file for variables to be wrote into and stored on that file.

An advantage of Binary Writer is that the file cannot be read in plain text if you’re accessing the file from outside the program so sensitive data is protected.

if (CurrentWave == -1)

{

wonGame = true;

//Adds to previous stats

player.EKilled += Game1.GAME.Leaderboard.enemiesKilled;

player.TLanded += Game1.GAME.Leaderboard.tearsLanded;

player.IPicked += Game1.GAME.Leaderboard.itemsPicked;

//Creates the leaderboards file

BnWr = new BinaryWriter(new FileStream("LeaderBoards", FileMode.Create));

BnWr.Write(totalWavesLasted + 7);

Writes the variables stored in the GameDisplay class to a filed called LeaderBoards when the wave equals -1

BnWr.Write(player.TLanded);

BnWr.Write(player.EKilled);

BnWr.Write(player.IPicked);

BnWr.Write(wonGame);

BnWr.Close();

Game1.GAME.CurrentDisplay = Game1.GAME.LoadingScreenDisplay;

}

## Collisions

### Rocks

A problem I came across while making the game was collisions. I used rectangles that were drawn around the player as well the rocks, detecting if both rectangles intersected each other. However sometimes the player would not be stopped by the rock so I came up with a solution to fix this.

#### Solution

Instead of using one algorithm to detect a collision I used abstraction to split up the code so I could account for different scenarios.

If the player is intersecting the bottom face of the rock, the player **CANNOT** be intersecting the right, left, top and diagonal faces

Rock

If the player intersects at the left face of the rock then stop the player’s motion. The player **CANNOT** be intersecting at any other face of the rock.

Rock

Devising these solutions allowed the collisions to work 100% as before this solution the rock would execute the wrong function and allow the player to walk through the rocks or teleport the player through the rock.

#### Algorithm

//Checks through each rock every frame and the player position

foreach (Rock rock in Game1.GAME.GameDisplay.rocks)

{

//Checks to see if the hitbox of the player intersets any of the rocks

if (hitBox.Intersects(rock.hitBoxR))

{

// Right side collision checks

if (hitBox.Right > rock.hitBoxR.Left && Position.X < rock.Position.X)

{

Position.X -= playerMoveSpeed;

}

// Bottom side collision checks

if (hitBox.Bottom > rock.hitBoxR.Top && Position.Y < rock.Position.Y)

{

Position.Y -= playerMoveSpeed;

}

// Left side collision checks

if (hitBox.Left < rock.hitBoxR.Right && Position.X > rock.Position.X)

{

Position.X += playerMoveSpeed;

}

if (hitBox.Top < rock.hitBoxR.Bottom && Position.Y > rock.Position.Y)

{

Position.Y += playerMoveSpeed;

}

}

}

### Enemy and Player Projectiles

The enemy and player collisions work the same as rock collisions work where both projectiles/tears have a rectangle drawn around their position which updates when the projectiles move. If the enemy tear intersects with the player the player is damaged and the tear disappears. If a player tear intersects with an enemy that player dies.

#### Algorithm (Enemy Tear)

//Checks to see if any of the enemy tears hit the player if so deduct health from the player

if (hitBox.Intersects(Game1.GAME.GameDisplay.player.hitBox))

{

Game1.GAME.GameDisplay.player.Health -= 40;

isDead = true;

}

#### Algorithm (Player Tear)

//Checks enemy to see if they interset with any tears

foreach (Entity e in Game1.GAME.GameDisplay.entities)

{

if (e is Tears) {

if (hitBox.Intersects((e as Tears).hitBox))

{

Game1.GAME.GameDisplay.player.TLanded++;

Health -= Game1.GAME.GameDisplay.player.playerDamage;

(e as Tears).isDead = true;

}

}

}

# Testing

## Proposed testing

Due to time constraints, I have decided to limit myself to only using white box testing as I can tweak the code if any bugs appear. This allows me to polish out the games bugs and errors so it is a perfect product.

Most of my data will be Normal data which is data accepted by the program, this is due to the user only being able to control a character using the arrow keys as well as shoot with WASD, furthermore the user can only click on buttons. Some Boundary data will be used to see if the game will “glitch” and not work as intended. For example, the player should only be able to spawn 2 tears at wave1.

## Testing Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test  Number | Description | Data Type | Expected Outcome | Outcome |
| 1. | Menu Displays when game is initialised | Normal | The menu displays when the game is launched |  |
| 2. | Click on the play button | Normal | When the play button is clicked the game directs to the character selection screen |  |
| 3. | Click on the play with cutscene button | Normal | Play with cutscene button directs user to cutscene |  |
| 4 | Press enter when the cutscene is playing | Normal | The cutscene will skip to the next slides in the cutscene |  |
| 5. | User picks player 1 | Normal | The player 1 character model will spawn and have player 1 attributes |  |
| 6. | User pics player 2 | Normal | The player 2 character model will spawn and have player 2 attributes |  |
| 7. | User presses WASD while the gamescreen is running | Normal | the character shoots tears in their corresponding direction |  |
| 8. | User presses arrow keys when the gamescreen is running | Normal | The player will move north if the Up key is pressed, East if the Right key is pressed, South if the Down key is pressed and West if the Left key is pressed. |  |
| 9. | User moves into the faces of the rocks | Normal | Collisions work with the player colliding with the rock’s faces |  |
| 10. | User moves the player into the corners | Normal | Collisions work with the player colliding with the rock at the corners. |  |
| 11. | Enemies shoot projectiles at set intervals. | Normal | The enemies shoot in diagonal velocities |  |
| 12. | Enemies move in the same direction and rebound off walls | Normal | The enemies reverse their velocity when they collide with the edges of the screen. |  |
| 13. | The user shoots the enemies | Normal | The enemies die after being hit a certain amount of times |  |
| 14. | The user allows the enemies to shoot them. | Normal | The player dies and the game exits. |  |
| 15. | The player moves over an item | Normal | The players texture changes and the item disappears |  |
| 16. | The item changes the players attributes | Normal | The player moves faster and can destroy environments. |  |
| 17. | The player shoots at the rock when they have an item | Normal | The rock’s texture changes and eventually gets destroyed |  |
| 18. | All enemies are killed by the user | Normal | The transition screen displays over the game display |  |
| 19. | The transition screen completes | Normal | The room changes texture into Room2 |  |
| 20. | Wave 2 initiates | Normal | Enemies spawn a swell as new rocks in a different formation |  |
| 22. | Waves 1-7 work with succession | Normal | After each room is cleared the rooms change and each wave spawns |  |
| 23. | Rooms specific items spawn | Normal | New items spawn at wave 1, 3, 4 & 5 |  |
| 24. | The player completes wave7 | Normal | The user is directed to the ending screen |  |
| 25. | The player initiates the game | Normal | The main menu song plays |  |
| 26. | The player loads into the Game Screen | Normal | The Game Display soundtrack plays |  |
| 27. | The user waits for songs to change | Normal | The next track in the Game soundtrack plays |  |
| 28. | Clicking around the boundaries of the main menu buttons | Boundary | The game will not redirect the user or Exit the game. It should stay on the main menu screen |  |
| 29. | Press the S key repeatedly in rapid succession | Boundary | The player will only shoot 2 tears down wards and no more until the tears have de-spawned. |  |

## Testing Video

LINK TO TESTING VIDEO

## Testing Outcome

# Evaluation

## Analysis Objectives Review

## User Feedback

I conducted an interview with the lead guitarist “Sylvester Volgaah” of the band Volniir on his views on the project as the project is being used as a marketing campaign for their new EP as well as is experience with the project.

I sent over a copy of the completed project and asked him to give feedback on the overall solution and what could be done better.

**“What did you think about the overall gameplay of Volniir?”**

Sylvester: The overall gameplay of the game was easy to pick up and learn. I liked how the character movement was fluid and dynamic being able to doge enemy projectiles but then having a chance of still getting hit by another enemy. I liked how the player was restricted to only shooting 3 times and not being able to spam the projectiles, it added an element of difficulty. It is definitely a hard game to play but is rewarding and fast paced. The enemies weren’t too hard nor too easy as well I liked how they varied each wave.

**“What did you think of the graphic design of Volniir?”**

Sylvester: Volniir is a beautiful game reflecting upon the dark theming of The Enigma EP. I loved being greeted with the bands logo infused onto some medieval art piece. The character and enemy design I thought represented the band’s corpse paint what we wear on stage, that was a nice touch. The design I loved the most was the room design and how it looked like a medieval dungeon and changed each wave into something unique but the same.

**“What did you dislike about Volniir?”**

Sylvester: I disliked how short the game was. Even though the game did increase in difficulty it only lasted 7 waves which was way too short. As well I thought there could have been more gameplay features implemented like a currency or a variety of weapons.

**“What could be improved with Volniir”**

Sylvester: As I understand this is being used as a marketing item for our EP. I thought that it wasn’t prominent enough that the game was intended for that. It needs to be more clear that it is a promotion for The Enigma. The game’s length needs to be extended and needs more variety to make it more of an interesting game.

### User Feedback Review

The project has defiantly achieved the objective of being a simple, fun and medieval style dungeon game. I am happy that the band are overall happy with the project and that they like the finished solution. However, I do agree that the project could be more clear that it is a marketing tool for the Band and needs more pictures and overall links to the EP inserted into the game, for example having a link on the main menu screen that sends you to the bandcamp website where you can buy the EP. Due to time constraints and limitations I wasn’t able to develop the game more but I do agree that the game could be longer and offer more features to make it more interesting and engaging.

## Review of overall system

The overall system is a polished game which offers hours of gameplay and fun as intended. It has a gritty art scheme what fits the theme of the game well and looks beautiful. The game has very little bugs and errors that need to be fixed and runs very well. Most objectives were met and implemented into the project as well as some other features that weren’t intended to be added initially. It could use further improvement in terms of variety and complexity to increase the overall play time of the game as well as further replay ability. It has met its objective as a marketing tool but that furthermore could be improved so the audience know what the game is for.

I am overall pleased with the final solution thinking that it is an enjoyable game that spreads awareness to people about the band Volniir and their latest EP The Enigma.

# Bibliography

Monogame Extension for Visual studio:

http://www.monogame.net/

Placeholder and foundation graphics: <http://store.steampowered.com/app/113200/The_Binding_of_Isaac/>

Soundtrack files: <https://www.youtube.com/watch?v=FH1ccqGIxPQ&safe=active>

<https://www.youtube.com/watch?v=1q1jbmbUHY0&safe=active>

<https://www.youtube.com/watch?v=UPrqyGbmsys&safe=active>

<https://www.youtube.com/watch?v=ro6jXIZiEfY&safe=active>

Animation algorithm:

<http://rbwhitaker.wikidot.com/monogame-texture-atlases-2>

Volniir (Band):

<https://volniir.bandcamp.com/releases>