National College of Ireland

BSc in Computing

2015/2016

Luke Stephens Kehoe

x15018849

Luke.StephensKehoe@student.ncirl.ie

Evasion

Technical Report

1 CONTENTS

2 EXECUTIVE SUMMARY

The gaming Industry has changed significantly. Mobile games have exploded in the last few years ranging from simple puzzle games such as Angry Birds and Candy Crush, to quite complex multiplayer games such as Game of War and Clash of Clans. It presents itself with a few clear advantages. Most people already own a smartphone. Without the need to purchase a gaming rig such as a console or high end PC, it has now become the most popular platform for gaming traffic. The rise of mobile games have attracted new customers to the industry that would not be associated with Gaming at all such as elderly people, teenage girls and the infamous "Candy Crush Housewives".

This part of the industry has taken off so much that successful studios such Irrational Games who released "*Bioshock* Infinite" on PC/Xbox 360/ PlayStation 3, to both critical and commercial success stated that they would not be working on a direct sequel. Instead, they would be venturing into mobile games, as it was much more lucrative. Even in mainstream media during the Super Bowl 2015, the only Video game to appear on the halftime show commercials was Clash of Clans. The Candy Crush Saga is valued greater than the Star Wars franchise.

The main objective of this project was to create a fun and addictive mobile game using Unreal Engine 4. The environment and all Game objects were created in this engine. The coding itself was done completely through Unreal Engines blueprints scripting, which is a visual representation of C++. All textures, materials and meshes used were provided by Unreal engines starter content and sample projects with the exception of a 'Stylish forest' that was purchased but does not appear in the final game. Some slight changes were made to various GUI elements and meshes using Maya and Photoshop.

3 PROJECT TIMELINE – INTEGRATIONS AND CHANGES

3.1.1 BLITZ

The project has gone through a number of changes and developments since it began. Initially the plan was to create a virtual reality (VR) first person shooter game called Blitz. Blitz would involve a basketball type dynamic along with wall running and traditional shooting mechanics. However, this idea changed for two reasons. One - there were numerous problems with the VR headset (Oculus Rift) for which there was little documentation available for troubleshooting. There was also limited access to the headset and much of the time was spent trying to operate it. At the time I thought the scope of the project was too large and challenging for the timeframe. Looking back at it now I wish I had persevered with this original idea, I know that with the knowledge that I have gained since that I could accomplish it.

The project goal was then changed to creating a fun and addictive game for mobiles. This is a market that is expanding rapidly and is primarily dominated by games developed in the unity game engine. I wanted to continue to use Unreal engine even though it is not normally used for this platform.

3.1.2 CATAPULT

The second game concept was called Catapult. The premise of the game was to defend worms that are on the ground from birds that are descending from the sky to eat the worms. The player would do this by launching cats into the air using a slingshot (or Catapult) and would accumulate score based on the amount of birds hit. It would take advantage of being a 2D game within a 3D environment by allowing the player to switch their viewpoint in order it hit birds they wouldn't normally be able to.

The game would challenge the user in terms of speed and accuracy while being put under increasing pressure. The graphics and visual style would be simple but retain a bright, colourful and visually appealing appearance. It was designed to appeal to a younger audience but would not be exclusive to that audience, much like angry birds it would be accessible to all ages.

The concept controls would be very simple. It was imperative that the game would be instantly understandable, a simple up/down motion to shoot and a left->right to change viewpoints. New users should be able to work out how to play the game with just their basic knowledge of real world physics. There were further plans to add different types cats and

birds that behaved differently (some are faster, heavier, shoots more than one cat at a time).

The project ran into a few serious issues after a while. The objective was to have the birds flying and landing on the worms from the sky. Unreal engine allows AI components to move around by using a Navigation bounds, which indicates on which surfaces the AI can move along. As there are no surfaces in the sky this is a problem. The only way of resolving this that I found would involve the birds following a pre-set path, which is not how they were desired to behave.

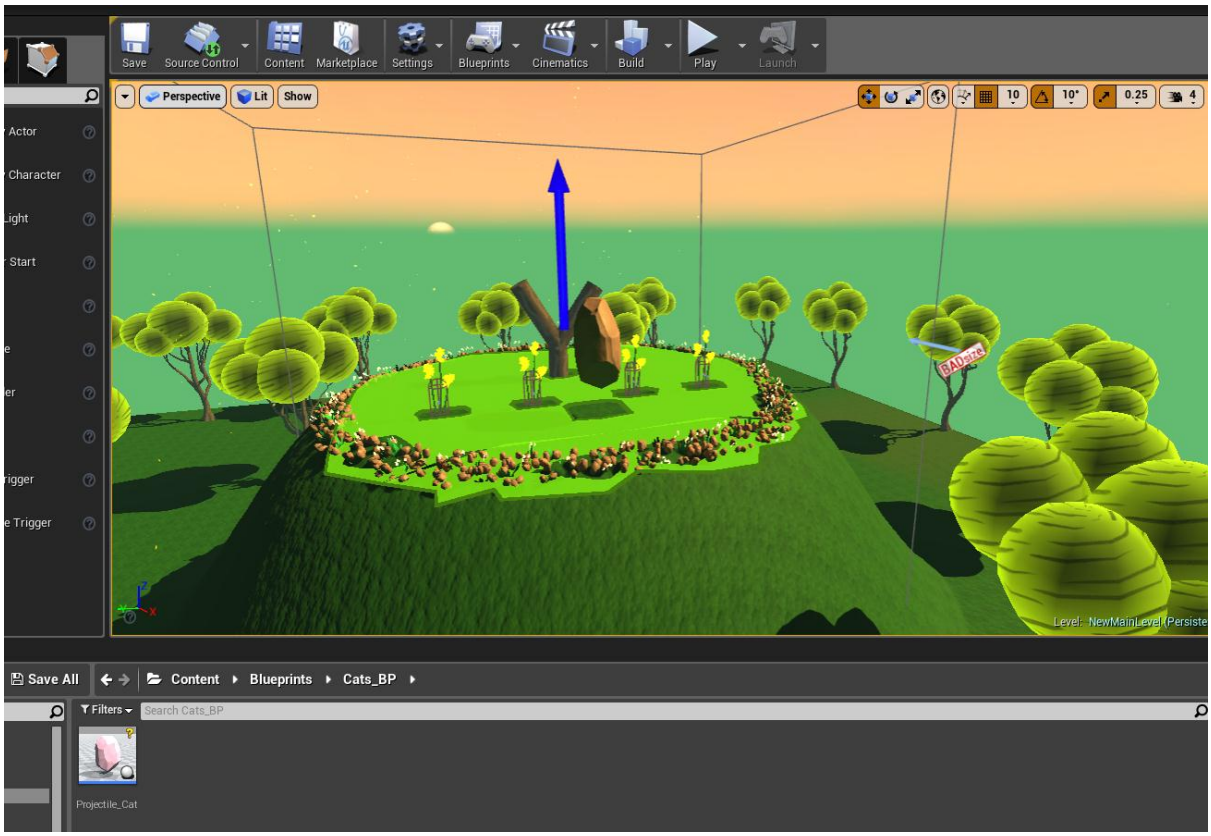


Figure 1 Catapult Version 1

This issue was resolved by lying everything on the ground and setting the camera to a Birdseye view. This solved the navigation issue but this broke the uniqueness of the multiple viewpoints. Another issue was that when spawning the projectiles (cat) the player would choose the x and y coordinates but the z would be inaccessible and would behave randomly. Overall the game was a bit of a mess!

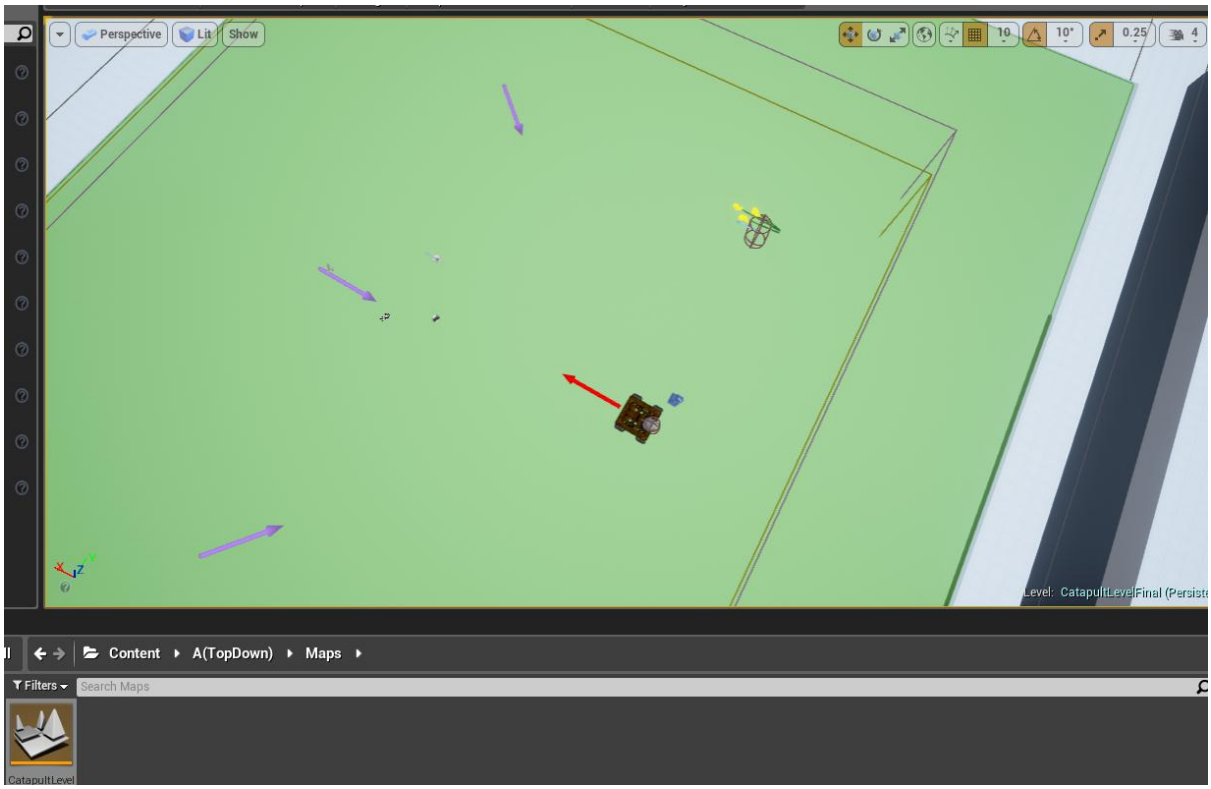


Figure 2 Version lying down, Birds eye view, green intended to be sky

3.1.3 EVASION

I decided to keep the assets, AI and game mechanics that had been successful but change the concept of the game once again. My final decision was to create 'Evasion', a game where the player has to pilot a ship around a small confined space and avoid enemies. The player can defeat enemies by hitting into explosive traps, killing the enemies in a small radius. The player is rewarded with a small amount of score but more importantly will drop three multiplier pick-ups. These will add to the player's multiplier, which multiples against any further score received by the player. The amount of enemies that will spawn will increase overtime and eventually the player will be overwhelmed and die.

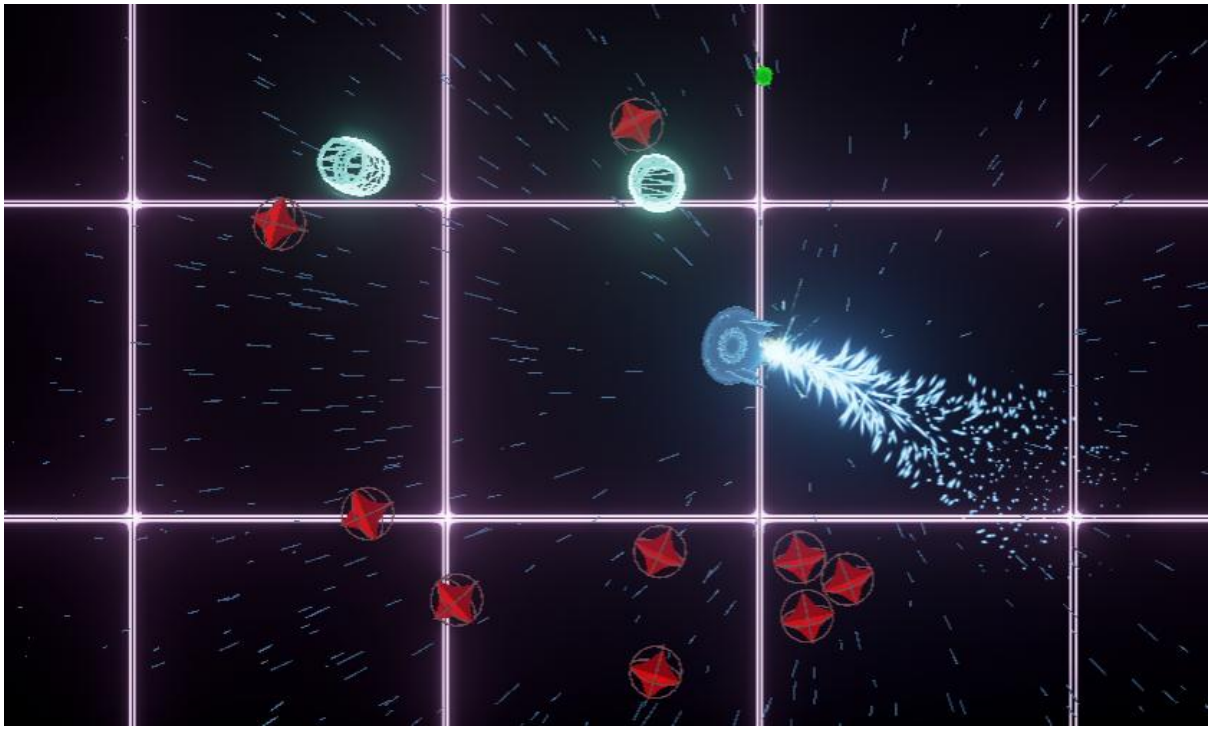


Figure 3 Evasion

Evasion is inspired by games such as geometry wars and Agar.io/Sliter.io. Geometry wars is a twin stick shooter developed for consoles. This game was ported to mobile, however, because it is a twin stick shooter it uses onscreen touch thumb sticks. These onscreen touch thumb sticks are not ideal because they are not very responsive. They also block much of the screen and they force the user to hold the phone at an uncomfortable angle. This made the game cumbersome to play on mobile.

As Evasion doesn't involve any shooting, the second thumb stick is not needed and so a better way of navigating the ship around would be to move with a single index finger similar to Agar.io/Sliter.io. This allows for the player to play one handed and at whatever orientation they want.

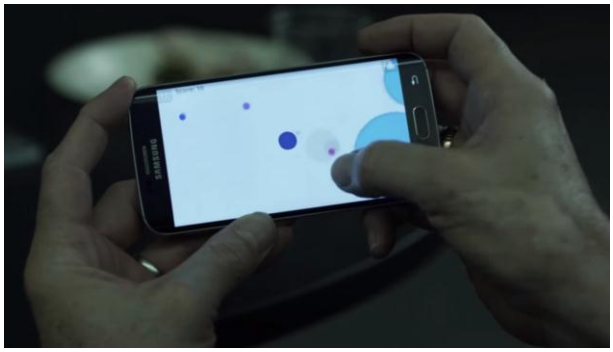


Figure 4 Agar.io as seen in house of cards

With the original idea '*Catapult',* it was intended to have a Cartoon forest look, but the styling that had been made did not suit the premise of the new game Evasion. This was changed so that the overall appearance would have a Neo-Arcade look. This was chosen because the meshes, textures and materials can be relatively simple so they do not impact game performance as much. Although it is attractive, the look has a few limitations as it may not attract very young audience and possibly female audiences. I have many ideas to appeal to these audiences by changing the art style to cartoon sprite based. See Further Development.

4 SYSTEM REQUIREMENTS

The requirements for this project have changed dramatically from what was previously outlined in the requirements specification as it has evolved to a completely different game.

4.1 FUNCTIONAL REQUIREMENTS

MAIN MENU LEVEL: this is default level. It is empty level where the player can press different buttons which link to different options. These options include the play (main level), view the high scores, options (sound), and quit game.

MAIN LEVEL: this is the main level where the player can play the game.

PLAYER MOVEMENT: The player should be able to move the controllable character. The player's character must move towards the user's finger (mouse pointer on computer).

RECORD SCORE AND MULTIPLIER: The game must be able to keep track to those two variables. When the player eventually dies they are presented with this score as a sort of recognition of how well they performed.

ENEMY DEFAULT CLASS: these are the main obstacles for the player. They must chase the player around the map and destroy the player upon hitting/overlapping with them. Additionally if the player is able to destroy them they must reward score and spawn three multiplier pick-ups.

ENEMY SPAWNER: there must be a way within the game that the enemies spawn according to the player's score.

TRAPS PICKUPS: these are pickups that when the player overlaps/hits these they must explode and destroy any enemies in a small proximity around them.

MULTIPLIER PICK-UPS: these are pickups that when the player overlaps/hits these they must add to the player's multiplier.

PAUSE GAME: The player should be able to pause the game and take a break from playing. The player can press a button in the top right-hand corner to pause the game.

QUIT GAME: The player should be able to quit the game and stop playing.

4.2 USER REQUIREMENTS

The user requirements are used to describe what the user will need in order to run and play the game. The game was designed with mobile devices in mind and tested against the Samsung galaxy s4. Thanks to Unreal Engine the game actually does support multiple platforms such as IOS, Windows, Linux, Mac, and HTML5.

The game is best played on Android devices. The user should only require a touch screen phone running Android (KitKat or higher).

4.3 ENVIRONMENTAL REQUIREMENTS

The environmental requirements outline the tools that were needed and used through-out the development of the project.

Unreal Engine 4 provided most of tools needed for the project. It was used to create the game mechanics and also the materials and textures used for the models and terrains. Unreal also provides a vast amount of props and static meshes for the models of the actors

in the game itself. Maya was used to for modelling of certain characters and effects. Photoshop was used to edit some UI elements and textures.

Internet access was required to access resources online such as tutorials, both written and in the form of YouTube tutorials. It was also a requirement to download the software needed to develop the project.

4.4 USABILITY REQUIREMENTS

The usability requirements outline the design process of the project, how it will look and how easy it is too use.

The game has an extremely simple interface using the in-built unreal UI editor. These include labels, buttons, etc. which were used to give the game a clean look.

The game must be first and foremost simple to control and easy to understand.

5 DESIGN AND ARCHITECTURE

Unreal projects uses blueprint visual scripting. This is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. This system is extremely flexible and allows for easy adjustments and design. Each and every component within the game has its own blueprint script that allows for functionality to be built in. Through the use of Blueprints, designers can prototype, implement, or modify virtually any gameplay element, such as: Games - set up game rules, tweak gameplay conditions, etc. **Players** - create variants with different meshes and materials or character customization. **Cameras** - prototype new camera perspectives or change the camera dynamically during play. **Input** - change the player controls or allow players to pass input to items. **Items** - weapons, spells, pickups, triggers, etc. **Environments** - create randomized props or procedurally-generated items.

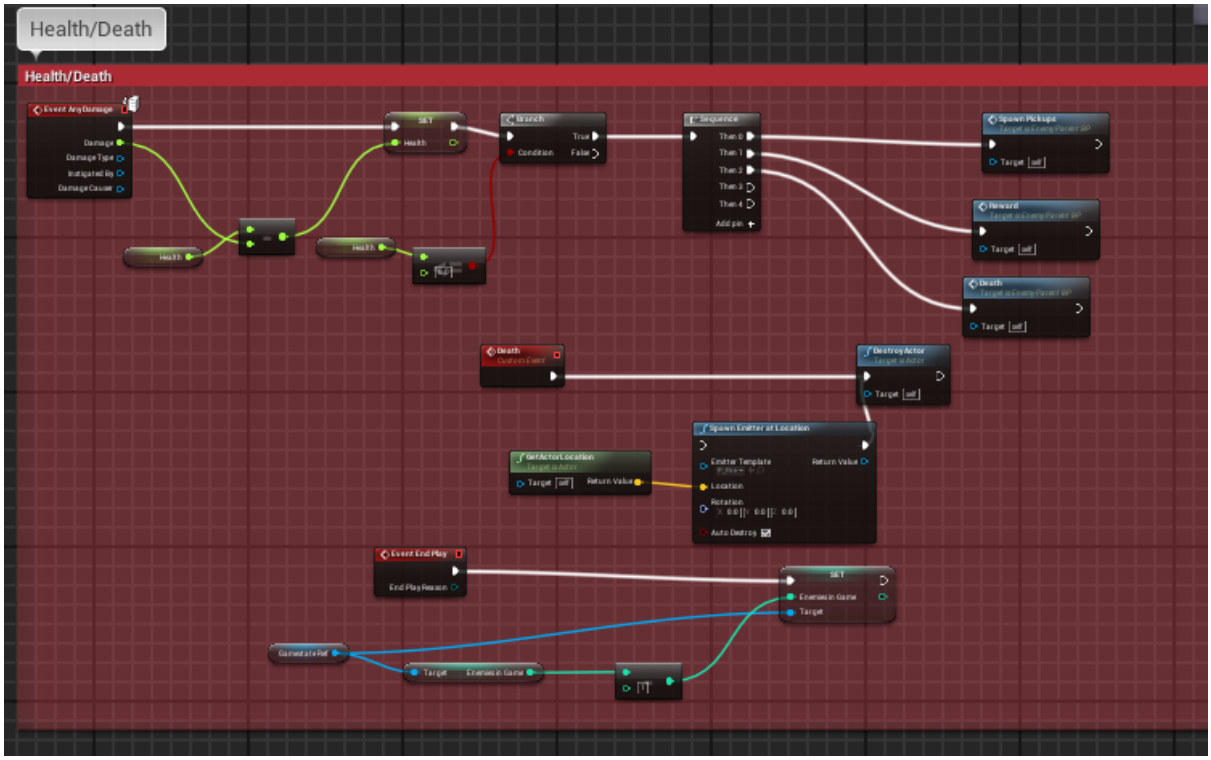


Figure 5 Blueprint Example

Each level of a game in Unreal has a number of properties which define what happens in that level. The level has its own blueprint script which allows it to perform functions in the game and set world properties such as gravity etc. It has its own world space in to which the developer can add actors, cameras, lighting, landscapes, terrain etc. into the game. Each level is linked to an independent game mode blueprint.

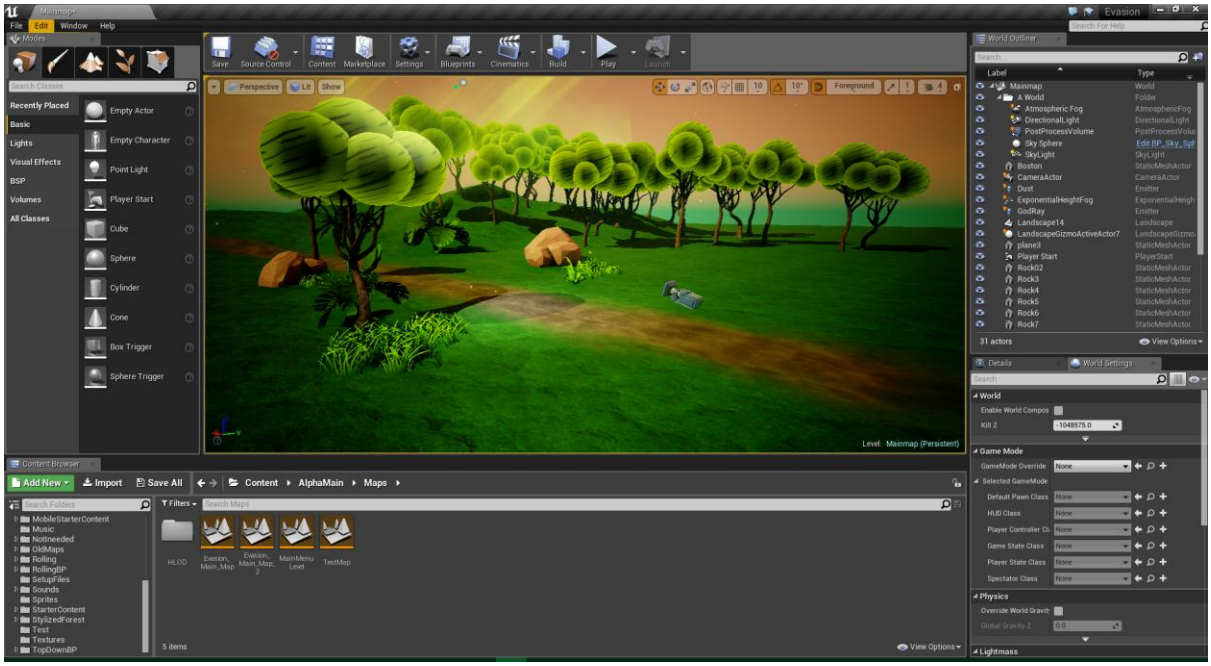


Figure 6 Example of editor

This game mode blueprint defines the default classes that will be used in the level. Pawn class, The HUD class, Player Controller class, Game State class, Player State class, and Spectator Class. Each of these classes are blueprints themselves and provide different functions within the game.

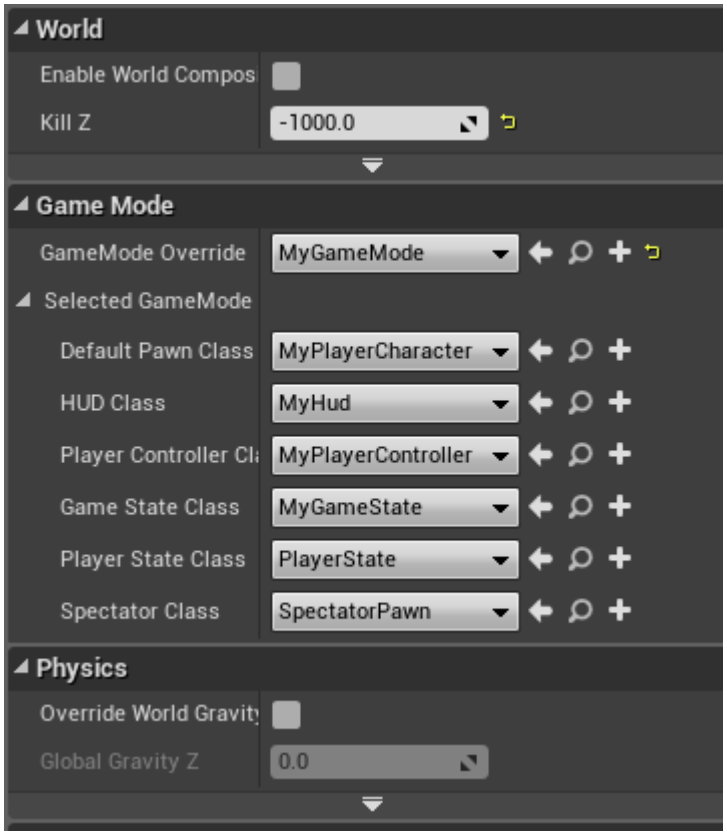


Figure 7 Level's Game mode tab

Default Pawn Class: Pawn that the player will control by default.

HUD: the heads up display that will show onscreen

Player Controller: class which controls the player's character

Game State: class which monitors the state of the game, score, list of players etc.

Player State: class which is used to replace the player controller in a network replication for all clients (default class will do for this project)

Upon creating a project Epic (Unreal's Company) provides developers with a large amount of starter content and a general template for the particular type of game that you wish to develop. This starter content includes meshes, materials, textures, basic character blueprints and an example map with those assets assembled. Epic also provides much free content that users can use in their projects.

6 USE CASE DIAGRAM

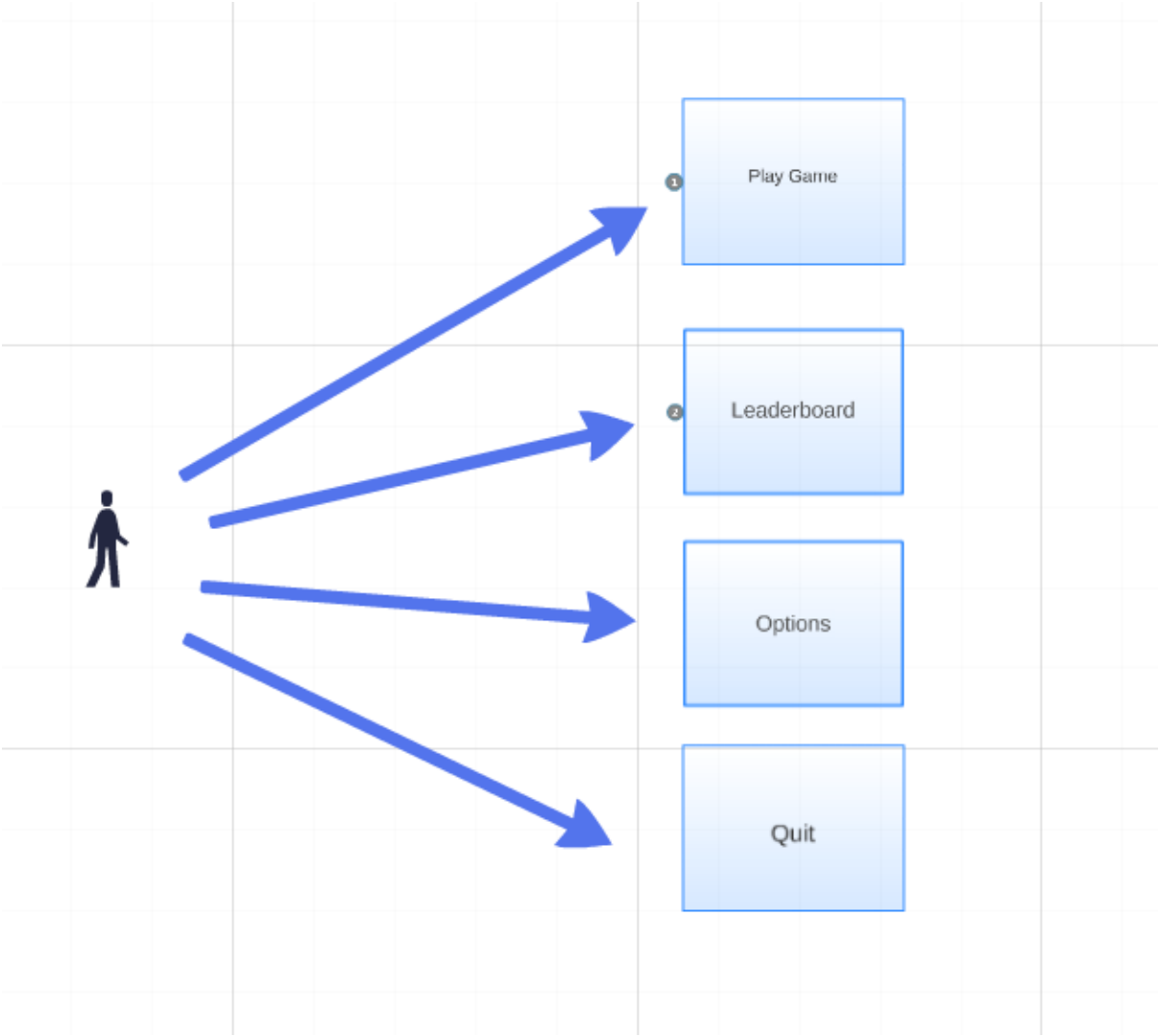


Figure 8 Case Diagram

When the player starts up the game the main menu will be displayed. The player will be presented with 4 options to choose from. The ‘Play’ option will bring the player to the endless main level and they will be able to play the game as normal. The ‘Leader board’ option will bring the player to a Screen which will display their high score. The options option will allow them to turn of the music. Finally, the ‘Quit Game’ option will allow the player to quit the game and exit the application.

7 IMPLEMENTATION

The project contains many blueprints but this is an overview of the main blueprints/classes that drive the entire game.

7.1.1.1 MYGAMESTATE_BP:

This class contains most of the variables that are used in the game. Nearly all other blueprints contain a reference to this class and pull information from it. It also updates the limit of enemies on screen in accordance with the score and the time that has passed. Every 7 seconds 3 extra enemies will be added to the limit and every 1000 of score 1 extra enemy will be added to the limit.

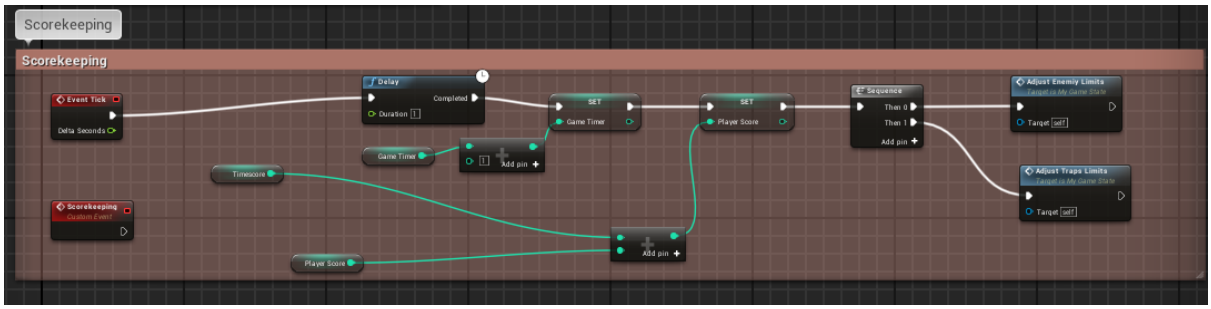


Figure 9 MyGamestate

7.1.1.2 MYPLAYERCONTROLLER_BP:

This is the class that handles input from the player. When the user presses with their finger (or left click) the player's actor will move to that exact location. It also disables input to the actor when the player's health is depleted.

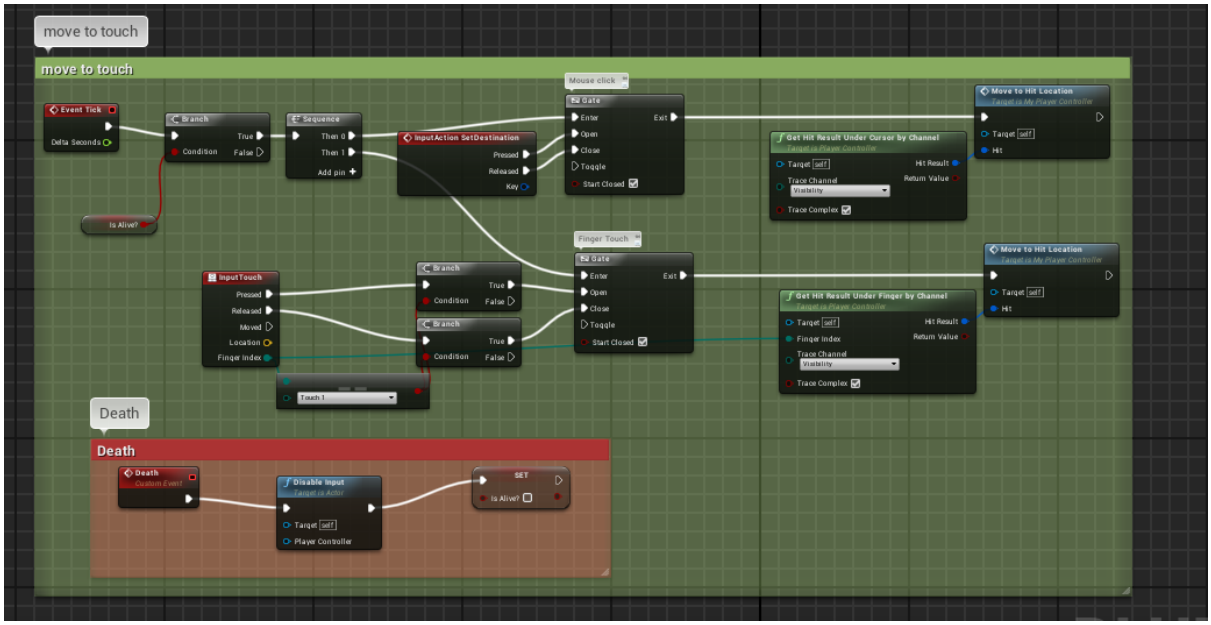


Figure 10 MyPlayerController_BP

7.1.1.3 MYPLAYERCHARACTER_BP:

This class contains the player's actor along with a few functions. The UI widget is called and displayed on the screen. The player's health is defined here and is set to regenerate over time.

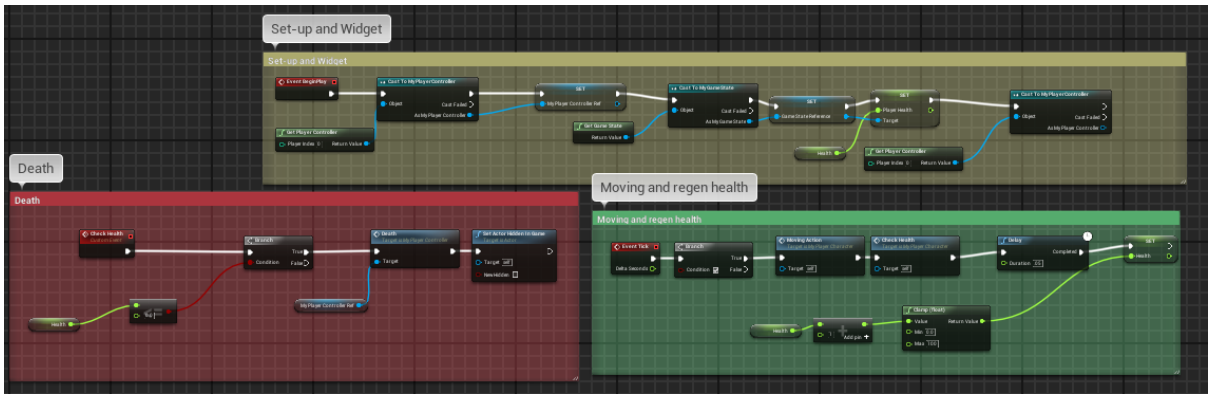


Figure 11 MyPlayerCharacter_BP

7.1.2 ENEMYPARENT_BP:

This is the parent class for enemies within the game. This is the original bird class that was made for catapult. The enemy when it spawns will delay for a quarter of a second to ensure the player has a fair amount of time to react to it. The enemy will then chase the player. By default the enemy is significantly slower than the player so the player can escape from them. On overlapping with the player it will deal damage to the player class and will self-destroy. If damage is applied to the enemy it will reward score to the player by calling the game state variable, it then spawns three multiplier pick-ups around its location. The amount of enemies in the game state is then reduced by 1 via calling the game state '*enemies in world*' variable, the actor is then destroyed.

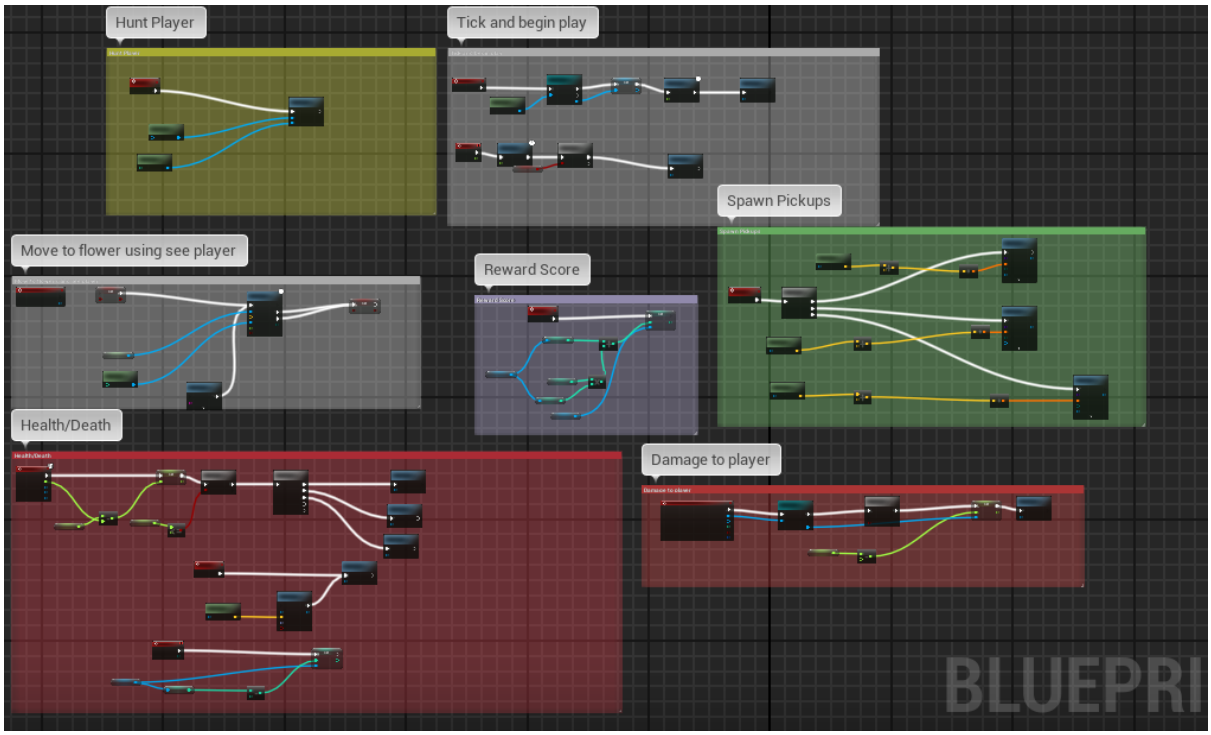


Figure 12 EnemyParent_BP

7.1.3 TRAPPARENT_BP:

This blueprint is an actor that acts as the player's weapon. When the player hits an actor of this class or any of its child classes they will explode and destroy any enemies in the area.

When one of these actors is touched by the player it reduces the amount of traps in the world by one (gamestate), releases damage out in a certain radius that the player is immune to, releases an emitter that looks like an explosion and deletes itself.

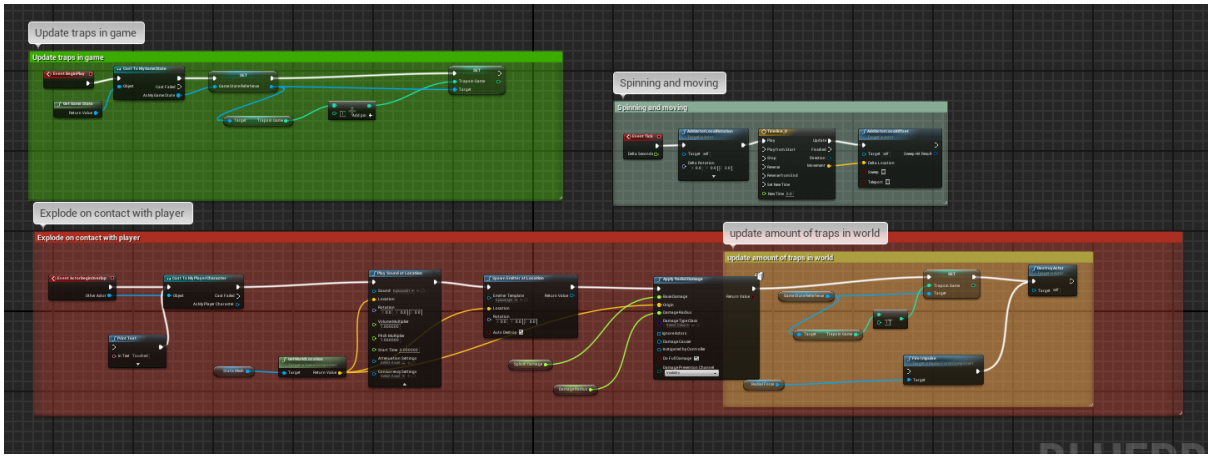


Figure 13 TrapParent_BP

7.1.4 ENEMYSPAWNER_BP;

This Blueprint is an actor that can be placed into a scene. It is hidden in game and will not intervene with the player. Its function is that every second it checks the amount of enemies in the level against the limit for the level. If the limit has not been reached it will spawn an actor from a list of enemies into the scene at a random point within a set radius within the navigable area.

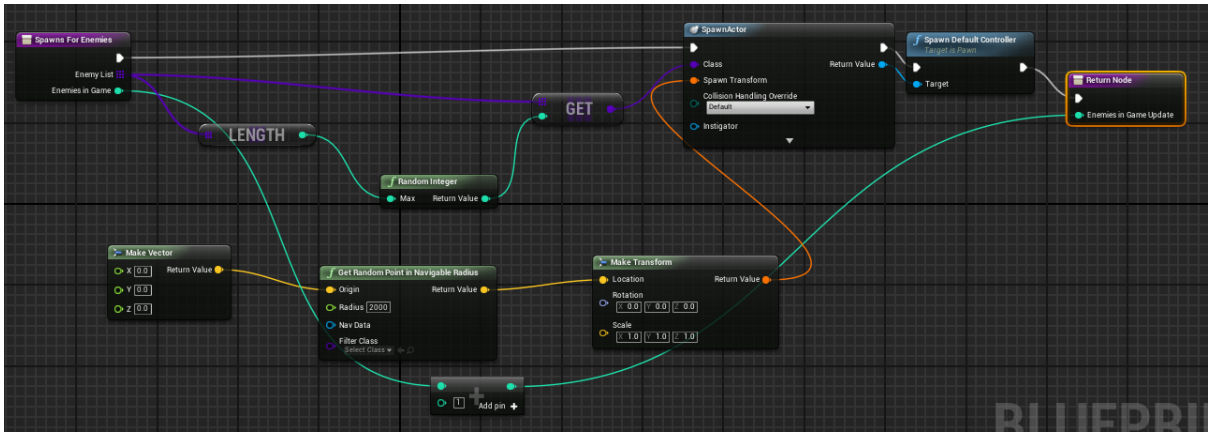


Figure 14 EnemySpawner_BP spawn function

The advantage of spawning enemies this way rather than using the level blueprint is that it provides greater flexibility. The spawner can be put into any level and will do the same function, allowing the creation of new levels very easy. The radius of the spawner can be adjusted in order to have a fixed spawn point. You can place multiple spawners into the level to increase the spawn rate very easily.

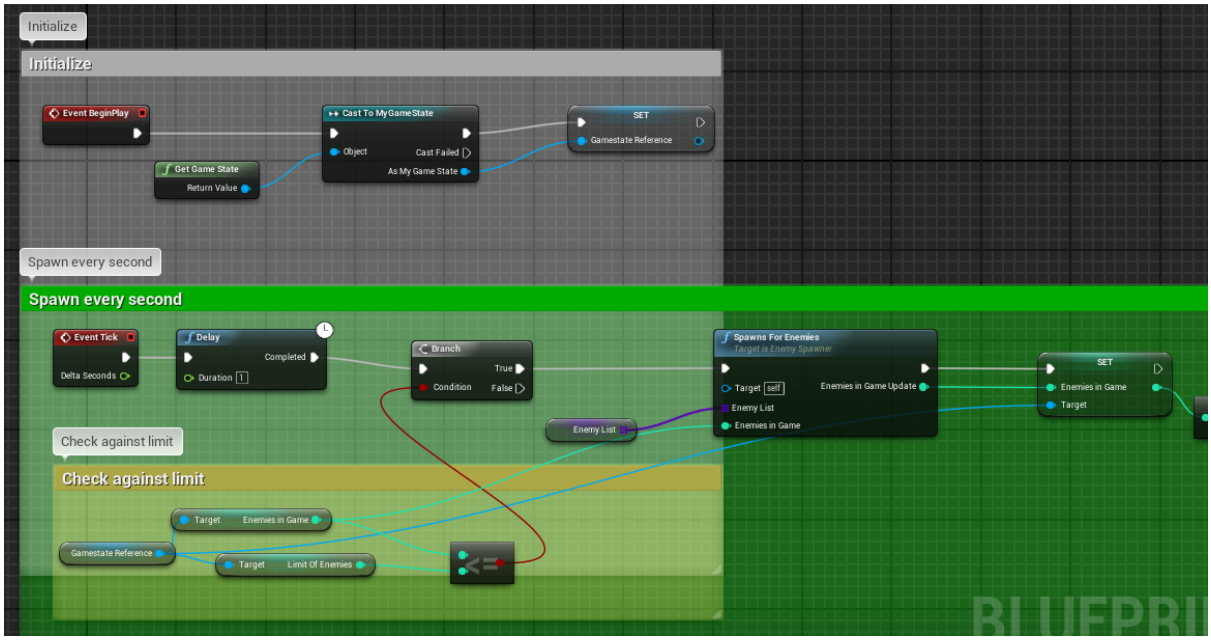


Figure 15 EnemySpawner_BP

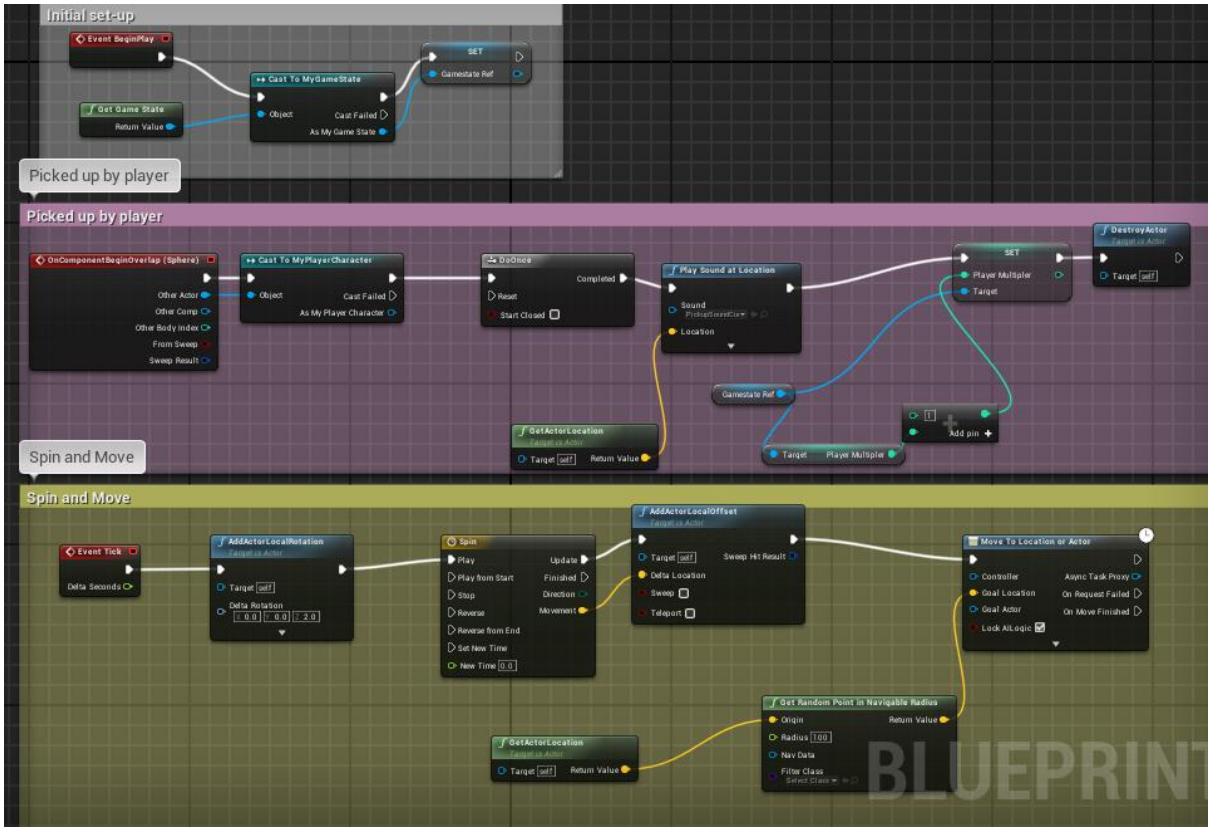
7.1.5 TRAPSPAWNER_BP :

This has the exact same functionality as the enemy spawner but with the traps. The limit is set much lower than with the enemies to ensure that the game remains difficult.

7.1.5.1 Multiplier Pick-up*:*

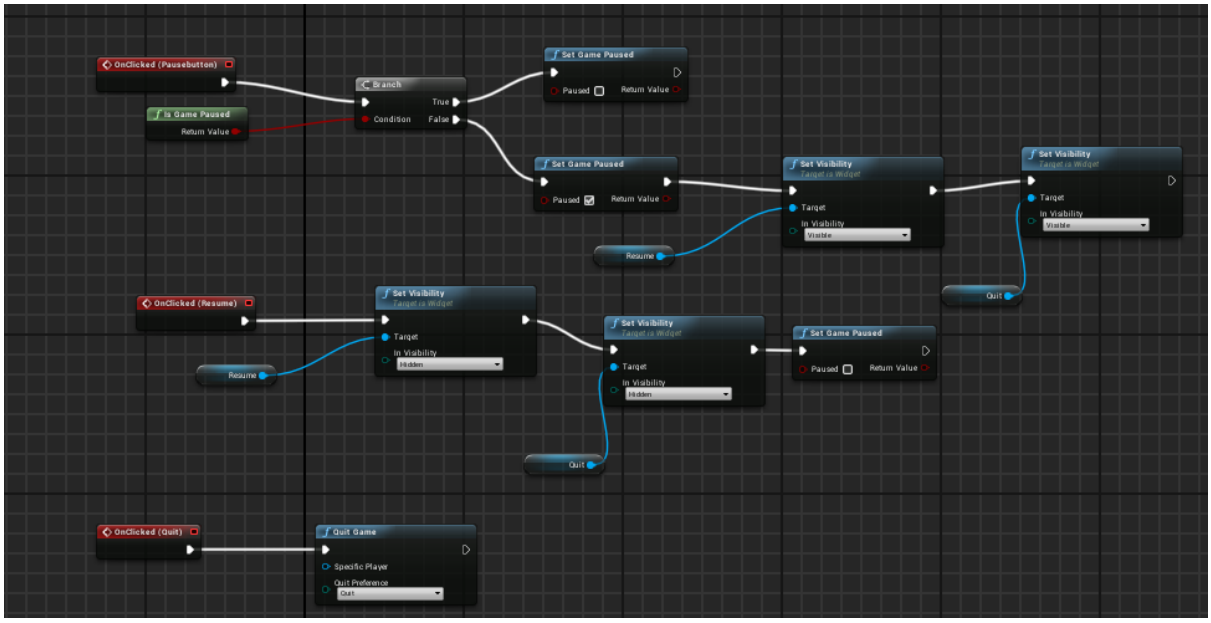
This is the Pick-up that is spawned after an enemy has been killed. It will self-destroy itself after a few seconds. Upon pick-up it will add one to the player's multiplier. It also contains

some code so it spins in place every tick (frame of gameplay) and plays a sound when picked up.



7.1.6 UIHUD_BP:

This controls the display of score, multiplier pause menu etc. Buttons hide and show components and pause the game. This is actually a widget rather than a HUD blueprint. I found the widget much easier to design and control then the HUD blueprint and achieved the same effect.



8 GRAPHICAL USER INTERFACE (GUI) AND AESTHETICS:

The cartoon meadow style that were created for the Catapult game did not suit the game mechanics of Evasion. As I am not a professional animator I did not have the confidence to create sprites that would look acceptable for the deadline of the project? I opted for the neo-arcade look after finding a tutorial of a material that produced a glow around the edges like the movie Tron. This material was perfect for the project as it used no textures, so it would have very little impact on the performance of the game.

This material was recreated and tweaked it to be transparent and was used as both a platform for the game stage, indicating the boundaries and also the background landscape. This allowed for the creation of simple but effective graphics that suited the style of the game. The main menu of the game is a copy of the main game's level but with a few moving objects to create an interesting effect and the camera is placed very far away. A widget is displayed over the level with the four options outlined in the use case section of this report.

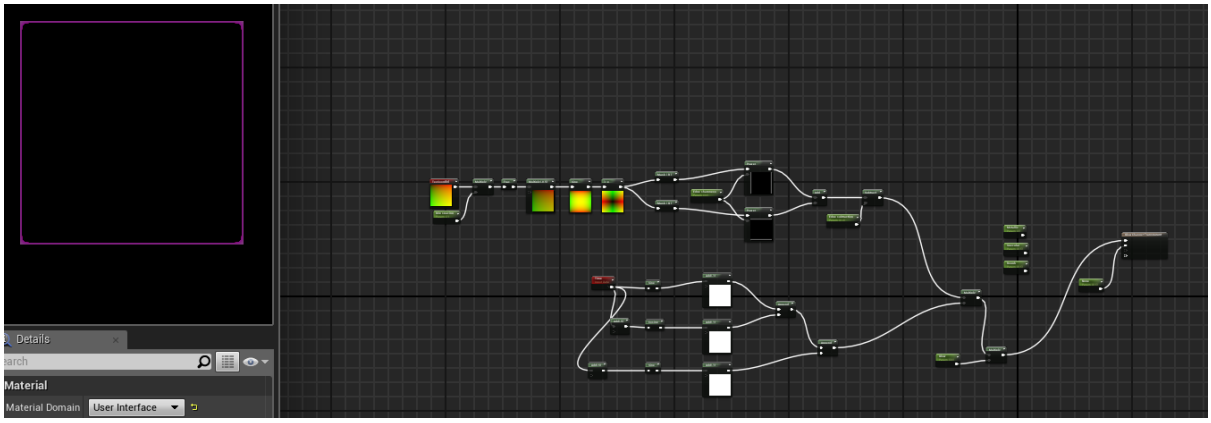


Figure 16 Material Creation

The player character was also given an emitter that shoots sparks from underneath the ship. This made the landscape look dynamic as the emitter would follow the player and appear to be part of the background.

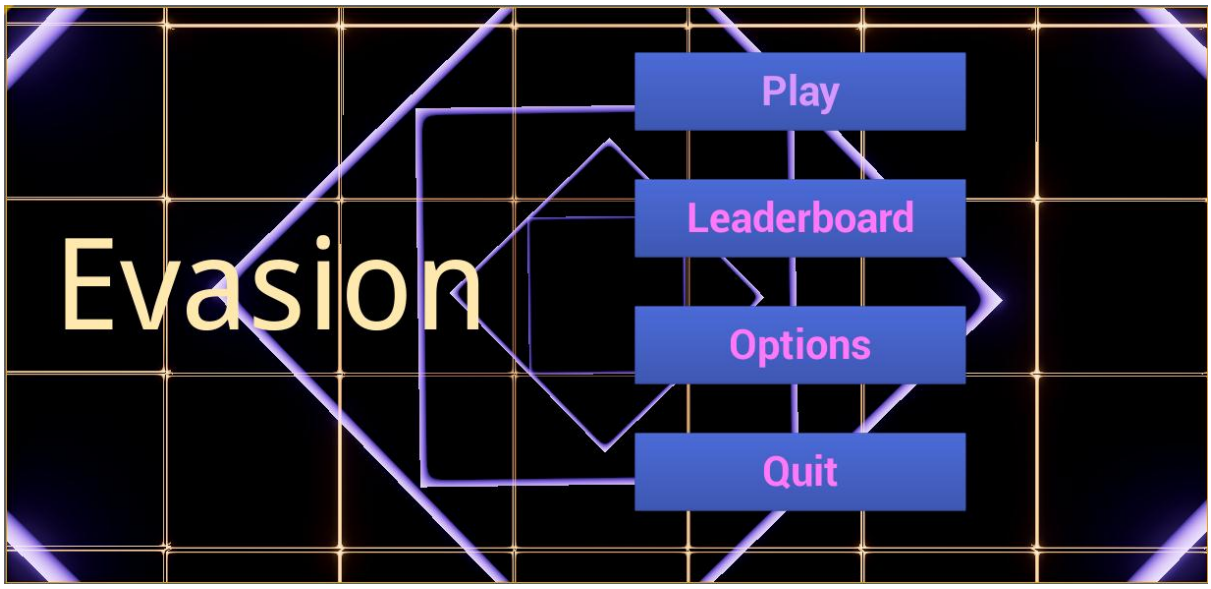


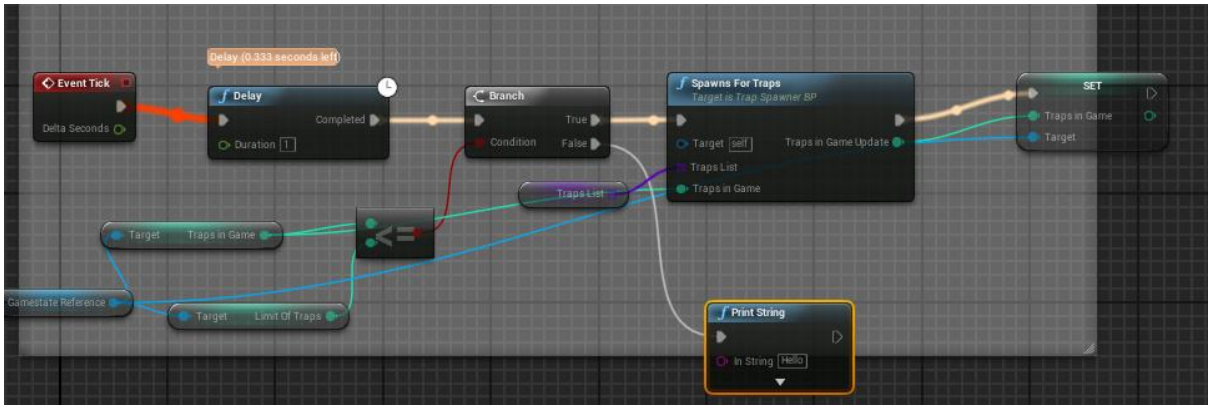
Figure 17 main menu

9 TESTING

This section will describe how the project was tested and evaluated.

This is an area where Unreal engine really shines. Unreal has an inbuilt debugger and console which can be used to monitor what the game is doing. What I found to be particularly useful was that unreal shows the flow of game through the nodes in each blueprint. This makes it very easy to see exactly what the game is doing at any given moment in real time. It allows you to drop in breakpoints at any particular node and inspect

the values, It also has a simulate mode that allows you to run game logic in editor viewport and inspect AI as the game characters perform actions.



The debugger tab allows developers to watch values for any specific actor. Additionally I tested the game on two mobile devices, Samsung S4 and S5. This insured that the game performed on the intended hardware. The game was tested by fellow classmates, friends and family, each giving feedback on the performance of the game and any changes that needed to be fixed.

10 CONCLUSIONS

The main reason for electing to make a game for my final project in that I have a strong interest in gaming and it's potential. Creating a game combines all the factors that interest me technology, visual appeal, strategy and imagination. Gaming is still a growing market which can be developed in many diverse ways. Having a genuine interest in this area meant that I enjoyed working on it and was more enthusiastic about completing the project. As well as the opportunity to create a game this project offered me the chance to learn and become proficient in Unreal Engine 4.

In my opinion Unreal Engine 4 is the best available (at no cost) game engine. It is widely used in the games industry so I wanted to become familiar with its use. I have had some experience using C++ the language that unreal is based on so I thought that I could use this knowledge. Doing this project has helped me think more analytically. The main disadvantage that I encountered was getting overwhelmed by the initial task. I had never

used a game engine before and I spent a lot of time trying to fix one particular difficulty which slowed down the progress of the project. I was forced to change and adapt my ideas and concepts. The positive outcome of this project was that I learned to adapt, to reconsider ideas and to find solutions. The completed game 'Evasion' is a relative success as I managed to achieve my objective of making a fun, mobile game and become very proficient at using Unreal and coding concepts.

11 FURTHER DEVELOPMENT OR RESEARCH

There are many ways in which the game 'Evasion' can be developed.

An immediate development that can be quickly implemented is adding new types of enemies that have different behaviours. New levels with different obstacles can be added to the game.

In order to appeal to various audiences the art style can be changed. For younger audiences the game could adopt a more sprite based 2d animated. A theme that could be adopted is to change the current spaceship to a boat in the sea, the enemies being sea monsters. For older audiences another concept is to have the setting in a bloodstream where the enemies are a virus.

Another intended development is to add multiplayer where up to 500 players can be on single map and they compete to be the top of the leader board, similar to slither.io. Once these changes are complete 'Evasion' will be published on the Google play store.

12 REFERENCES

https://docs.unrealengine.com/latest/INT/Videos/

https://www.youtube.com/watch?v=KHiZfy5OlO8

https://www.youtube.com/user/TeslaUE4

http://opengameart.org/

http://unrealtutorials.com/

Luke Stephens Kehoe

lukestephenskehoe@student.ncirl.ie

15018849

Catapult

Game Concept & Design Document

13 EXECUTIVE SUMMARY

Mobile games have replaced the consoles and PCs as the most populous platform for gaming traffic. The platform excels at bring users a fun and engaging experience while on the go.

Without the need to purchase a gaming rig such as a console or high end PC, the platform has attracted new users. The objective of this project is to create a fun and addictive 2d shooting game for mobiles in which the user must defend objectives.

The game will challenge the user in terms of speed and accuracy while being put under increasing pressure. It will be developed on a windows pc in Unreal Engine. The game will be developed primarily for Android based phones/tablets as they have the largest user base. The game could also appear on the windows store and may be ported to IOS at a later date.

14 INTRODUCTION

14.1.1 BACKGROUND

The gaming Industry has changed significantly. Mobile games have exploded in the last few years ranging from simple puzzle games such as Angry Birds and Candy Crush, to quite complex multiplayer games such as Game of War and Clash of Clans. It presents itself with a few clear advantages. Most people already own a smartphone. Without the need to purchase a gaming rig such as a console or high end PC, it has now become the most popular platform for gaming traffic. The rise of mobile games have attracted new customers to the industry that would not be associated with Gaming at all such as elderly people, teenage girls and the infamous “Candy Crush Housewives”.

This part of the industry has taken off so much that successful studio such Irrational Games prior to the release of “Bioshock Infinite“ on PC/Xbox 360/ PlayStation 3, which as both a critical and commercial success stated that they would not be working on a direct sequel. Instead they would be venturing into mobile games, as it was much more lucrative. Even in mainstream media during the Super Bowl 2015, the only Video game to appear on the halftime show commercials was Clash of Clans.

15 AIMS

The objective of this project is to create a fun and addictive 2D shooting game for mobiles in which the user must defend objectives. The game will challenge the user in terms of speed and accuracy while being put under increasing pressure. Given that this game will be expected to run on entry-level phones it is import that the graphics be simple but retain a bright, colourful and visually appealing appearance. The game is designed to appeal to a younger audience but will not be exclusive to that audience, much like angry birds it will be accessible to all ages.

The controls will be easy to learn and play, with very simple controls.

It is imperative that the game be instantly understandable. New users should be able to work out how to play the game with just their basic knowledge of real world physics.

The premise of the game is to defend worms that are on the ground from birds that are descending from the sky to eat the worms. The player does this by launching cats into the air using a slingshot (or catapult).

.

16 TECHNOLOGIES

Unreal engine 4 is a game engine which allows for development of 2D or 3D games for a variety of platforms. Unreal Engine technology powers hundreds of games as well as real-time 3D films, training simulations, visualizations and more. Unreal engine 4 allows you to browse C++ functions directly on game characters and objects by connecting to Microsoft visual studio in order to make changes.

Unity is another option that was considered as it is more suitable for 2d games, however Unreal is being used as it is one of the main learning outcomes of the project. Unreal uses

blueprint visual scripting which comes with a built-in debugger, allowing for interactively visualise gameplay flow and inspect property values while testing the game.

17 REQUIREMENTS

The game will only feature the Endless Mode and three or four other levels. Endless will include all content in the game while each other level will concentrate on just a single enemy/ power-up.

17.1 FUNCTIONAL REQUIREMENTS

Functional requirements outline the functions that the product must be able to perform.

Play Tutorial Level: This will be a short level that will explain the controls: how to use the slingshot and how to change the position of the camera.

Play Endless: This will be the main level of the game. This will be where all types of enemies, power-ups etc. will appear and is only ended when they have lost all their objectives.

Level Select: The player should be able to play and complete the level that they have chosen to play from the level select screen. The level selected will be loaded and the player is free to play through the level.

Complete level - The player should be able to complete the level and receive some sort of recognition from the game. With the exception of the classic mode the player should complete the level once they have held out for a certain amount of time (two minutes for example). When the player walks reaches that goal a box will pop up showing the stats from level and giving the player options to restart the level or return to the level select screen.

Pause Game: The player should be able to pause the game and take a break from playing. The player will press an icon at the top of the screen which will pause the game and give the player the option to quit the level.

Quit Game: The player should be able to quit the game and stop playing.

Fire Projectiles/Use Slingshot: The player must be able to fire the projectiles/ use the slingshot.

Fight/Defeat Enemies: The player should be able to hit/destroy and accumulate score from hostile enemies in the game.

Lose Objectives: This the punishment from missing any enemies. Losing all four will result in the ending of the game or failure of the level.

Save/Store High Score: It the player gets a score that is higher than their current high score then it will be updated

17.2 USER REQUIREMENTS

The user requirements are used to describe what the user will need in order to run and play the game.

From the user's perspective all that is required is an android touchscreen phone. The game will be developed with the mind-set of reaching as many devices as possible. This may change if the game is ever ported to windows or IOS.

As stated earlier it is imperative that the game be instantly understandable. New users should be able to work out how to play the game with just their basic knowledge of real world physics. No prior experience such be required and no tutorial should be necessary.

17.3 USABILITY REQUIREMENTS

The usability requirements outline the design process of the project, how it will look and how easy it is to use. The game has a simple interface using the built-in Unreal GUI elements. The game must be easy to use and understand, even to those who don't play games very often, if at all.

The game expected to be played using a resolution setting of 1080 x 1920 the resolution of high-end mobile devices. This will be downscaled if needed for lower end phones. The games GUI has been designed with this size in mind and look consistent when using this size. This all handled in Unreal.

DESIGN DOCUMENT

CATAPULT

18 INTRODUCTION

The objective of this project is to create a fun and addictive 2D shooting game for mobiles in which the user must defend objectives. The game will challenge the user in terms of speed and accuracy while being put under increasing pressure. Given that this game will be expected to run on entry-level phones it is import that the graphics be simple but retain a bright, colourful and visually appealing appearance. The game is designed to appeal to a younger audience but will not be exclusive to that audience, much like angry birds it will be accessible to all ages.

The controls will be easy to learn and play, with very simple controls.

It is imperative that the game be instantly understandable. New users should be able to work out how to play the game with just their basic knowledge of real world physics.

The premise of the game is to defend worms that are on the ground from birds that are descending from the sky to eat the worms. The player does this by launching cats into the air using a slingshot (or catapult).

There will be different types cats that you can launch that will have different properties (some are faster, heavier, shoots more than one cat at a time). Likewise there will be

different types of birds that will descend that will have different properties.(size, shape, speed) which will make the game more visually appealing and challenging.

Score will be accumulated by hitting the birds and hitting multiple birds with a single cat will increase the multiplier and award more score. The duration of the game is dependent on how quickly the birds eat the worms. The player's objective is to get the highest score possible before all worms disappear (eaten).

19 GAME ANALYSIS

Game Description	
Genre:	• Mobile, Neo-Arcade, Endless Runner/Shooter
Game Elements:	• Shooting, Puzzle, Speed, Accuracy, High Scores

Game Content:	• *Humour, Cartoon*
Theme:	• *Cartoon, Animated*
Style:	• *Cartoon, Animated*
Game Sequence:	• *Score will be accumulated by hitting the birds and hitting multiple birds with a single cat will increase the multiplier and award more score. The duration of the game is dependent on how quickly the birds eat the*

	worms. The player's objective is to get the highest score possible before all worms disappear (eaten).
Player:	● *The Number players that can play the game at once*
Game Reference	
Player Immersion:	● *Strategy, Mental, Reflex*

Game Technical	
Technical From:	● The gameplay will be on one plane 2D, The graphics will be 3D
View:	● Camera will be looking at the slingshot with birds descending from the top of the screen ● An additional feature may be that the player can re-orientate himself in the playing field (as the player's input is in 2D but the game is in 3d the player can swipe to adjust his shooting position)
Platform:	● Unreal Engine 4, C++,
Device:	● PC, Mobile
Game Sales	
Consumer Group:	● Younger players, All players

Payment:	• Advertisment
Estimated Price:	• Free
Device Support List	• Android Mobile Devices

19.1.1 GAME ATMOSPHERE

The game will be a cartoon animated. It will project a fun friendly environment that will attract a younger audience but not exclusively.

19.1.2 GAME PLAY

The gameplay will be fast-paced and frantic as players will struggle to cope with the volume of birds that will eventually be falling. There will also be an element of surprise and variation as the cats and birds will have different properties e.g. (weight, size, speed, flight path)

19.1.3 KEY FEATURES

- Number of Levels - One main level which will have all elements of the game in it and several other levels where there are only a set number of elements (eg only one type of Bird)
- Number of Enemies/ Characters - The game consists of 3 main characters, birds, worms and cats. There will be about 4 different types of birds and 4 different types of cats.

- Time of Game Play - average game length will be around 1 minute and then the player will replay to achieve a personal best
- Replay ability - to achieve a high score
- Audio Specifications - sound effects relating to characters
- Graphic Specifications - animation, low polygon count as it is running on a phone
- Device Compatibility - Android touch phones possibly IOS in the future.
- Number of Players - one only
- Online Activities - high scores which are comparable to friends
- Number/Type Modes - One main mode and one for every level

19.1.4 SELLING FEATURES

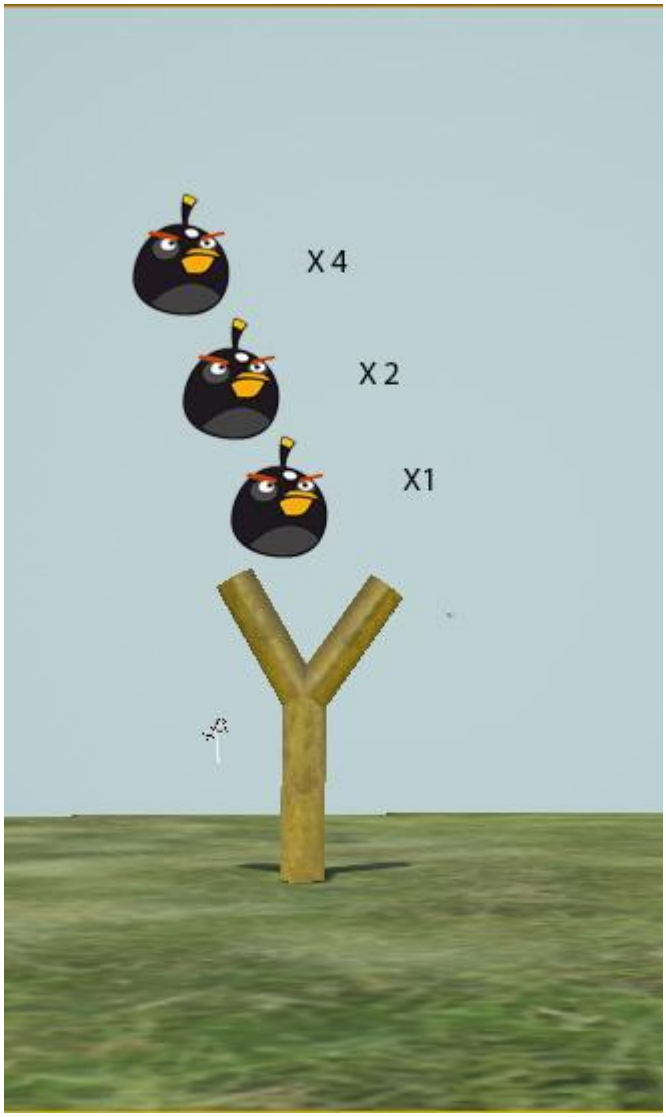
This is a fun easy to pick up and play game. It is ideal for small intervals of time when the person is looking for amusement. The game length is very short, designed to so it can be played at any given moment of the day.

19.1.5 GAME DESIGN DEFINITIONS

- Menu - Level Select, Endless mode, View High Score, Quit Game
- Game Play - Touching the screen and launching cats at birds midair to prevent the birds from landing and eating worms
- Player Control - Touching the screen and launching cats
- Game Over - All worms are eaten and a score is given

19.1.6 PLAYER REWARDS (POWER-UPS & PICK-UPS)

Build up score by hitting birds, and extra points are rewarded for hitting multiple birds with a single shot.

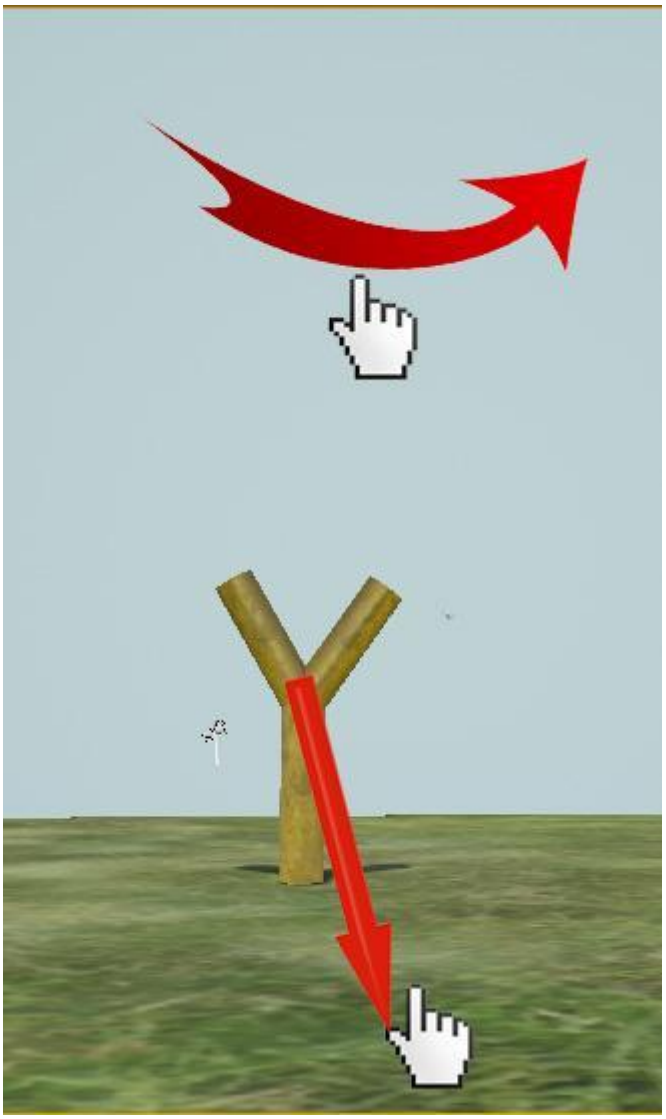


19.1.7 PLAYER PROPERTIES USER INTERFACE (UI) (CONTROLS)

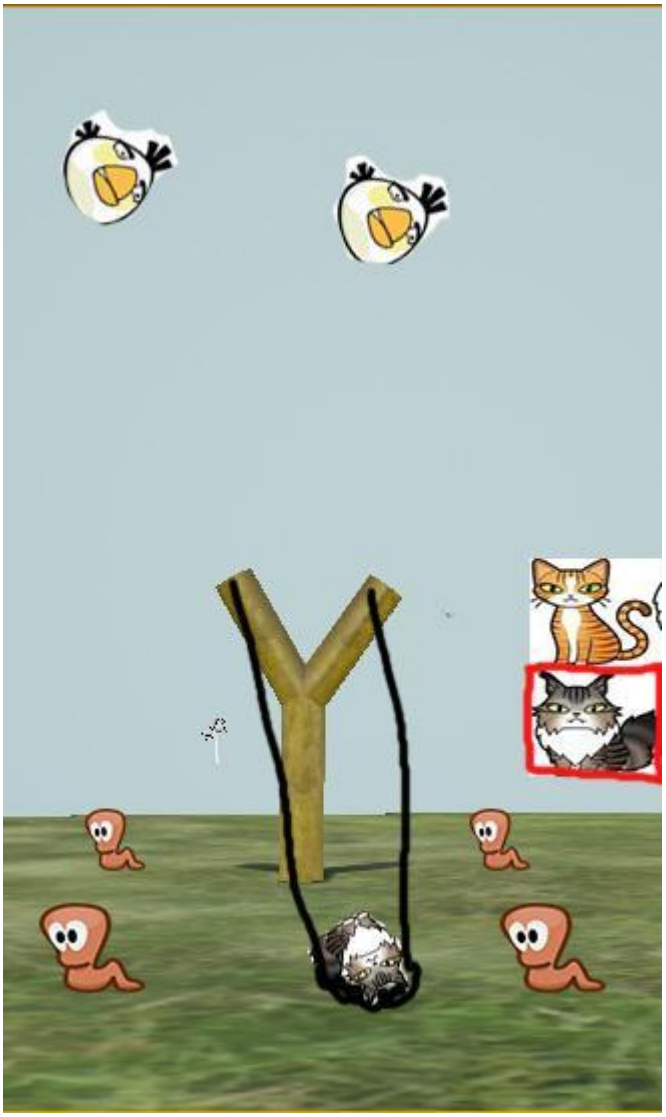
Move rotate camera, shoot slingshot

the player can re-orientate himself in the playing field(as the player's input is in 2D but the game is in 3d the player can swipe to adjust his shooting position) This may look unrealistic

because the birds will be flying in the exact same x and y coordinate but I intend to find a solution to this.



19.1.8 HEADS UP DISPLAY (HUD) PLAYER VIEW



19.1.9 ANTAGONISTIC ELEMENTS

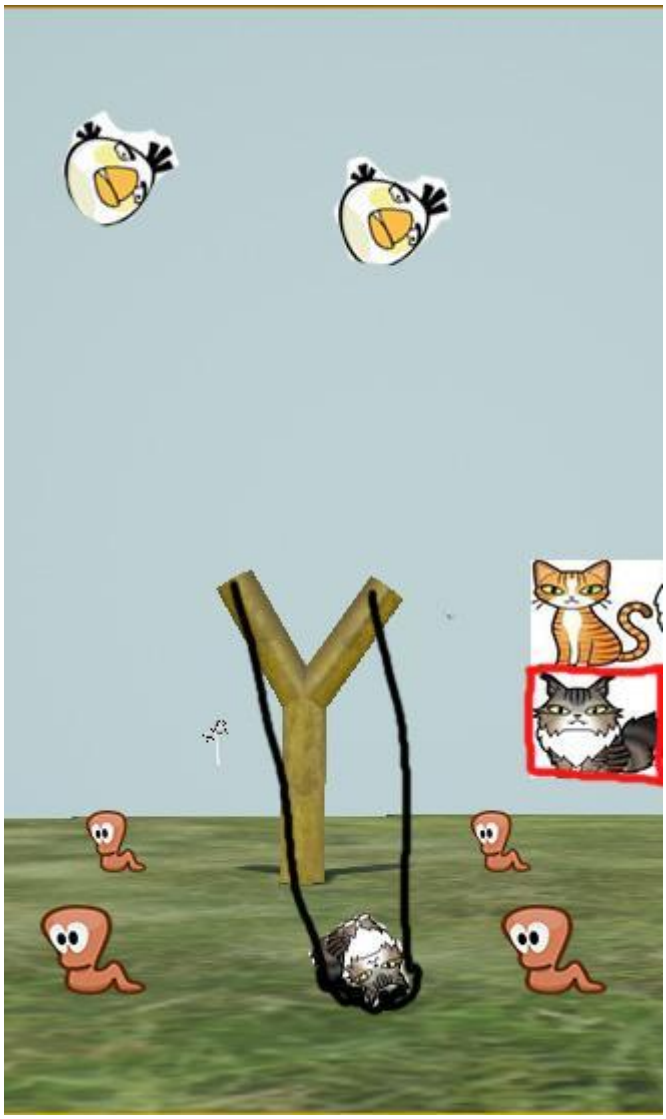
Four different types of bird to be designed in animation. They will vary in appearance and properties such as flight path, speed, weight etc. which will impact gameplay. All birds will target a worm to destroy and the player must prevent this.

19.1.10 ANTAGONISTIC PROPERTIES

All birds will target a worm to destroy and the player must prevent this. The birds will vary in their flight path, speed and weight.

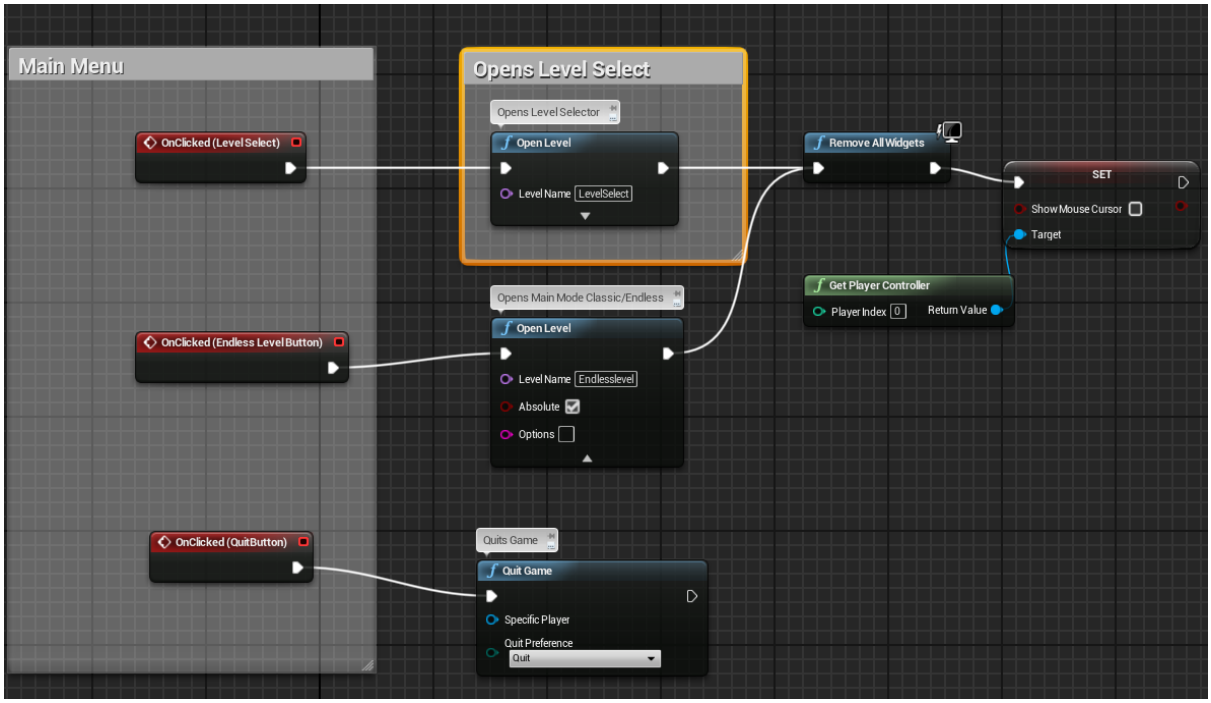
19.1.11 CONCEPT ART

Characters to be designed/ modelled in same style as each other. This will be done using Photoshop, Maya and unreal to produce the textures, materials and static meshes



19.1.12 GAME ARCHITECTURE

This is the preliminary main menu Blueprint, detailing the flow of the main (subject to change)



19.1.13 SYSTEM REQUIREMENTS

An android touchscreen phone is all the player needs

19.1.14 VISUAL CONTENT

Design of birds, slingshot, worms, cats all to be designed in a simple cartoon manner. Design of birds: four variations in colour,size and facial characteristics

Design of Cats: four Variations in colour, size, shape and facial characteristics

Design of worms will be all the same.

All characters will be designed in the same style. This will be done using Photoshop, Maya and unreal to produce the textures, materials and static meshes

19.1.15 AUDIO CONTENT

General background music - upbeat

Sound effects for birds, cats and catapult.

19.1.16 PROGRAMMING CONTENT

Events will be based on the blueprint scripts for each of the actors in the game (birds, cats etc.)

19.1.17 CONCERNS AND ALTERNATIVES

Input will be difficult to track as the player has no way of controlling the z-axis using the touch controls

19.1.18 RESOURCES

Unreal tutorials and documentation https://docs.unrealengine.com/latest/INT/

The game expected to be played using a resolution setting of 1080 x 1920 the resolution of high-end mobile devices. This will be downscaled if needed for lower end phones. The games GUI has been designed with this size in mind and look consistent when using this size.

20 PROJECT PROPOSAL – SLINGSHOT INVADERS (CATAPULT)

Project Proposal

Mobile Game - Slingshot Invaders

Luke Stephens Kehoe, x15018849,
x15018849@student.ncirl.ie

BSc (Hons) in Computing

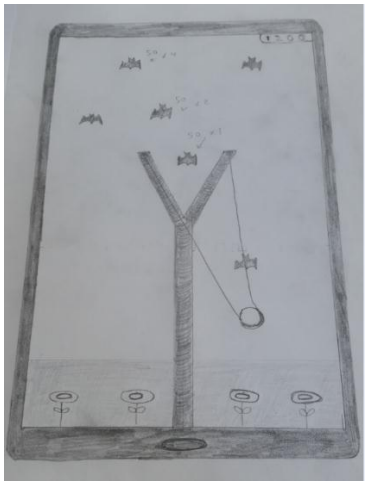
Gaming and Multimedia

28/09/2015

21 OBJECTIVES & BACKGROUND

The Objective of this project is to create a 3 dimension graphical mobile game that is played on a 2d plane. The game will be created for android initially and main game mechanic will involve a slingshot that the user controls to defend four objects (to be decided) by shooting projectiles into the air at the invaders (bats probably) that are flying down.

Depending on how far down the screen invaders are the amount of score the user will be rewarded, additionally if multiple invaders are hit with a single projectile there will be a multiplier that increases at 2^N rate 1st = x1, 2nd = x2, 3rd = x4, 4th = x8, …… capping at x32.



As the game goes on the amount of invaders that appear and the rate at which they fall will increase at every 5 - 6 seconds. Ideally the median game will last about 30 – 40 before the user can't deal with speed and volume of the invaders and is overwhelmed. His high score is recorded and hopefully the game was fun and addictive enough for them to play again

As it is going to be developed for mobile devices I will also like to incorporate leaderboards to display the player's high score with others. Facebook integration may suitable for this. Additionally different types of invaders and projectiles can be added into the game later.

22 TECHNICAL APPROACH

The game itself will be built in Unreal engine because of it superior graphical capabilities and I have experience with the engine. The engine has an inbuilt template for mobile development as well and there are many online tutorials on YouTube and Epic's website for support along with very active forums and subreddit.

The first milestone will be to create the game stage and get the core mechanic i.e. the slingshot working. After that is working the AI for the bats/invaders must be done. The invaders must try to destroy the flowers and therefore fall aiming towards them. It is also important that the path they take is somewhat unpredictable, as if they all follow a straight path the game will devolve into just "shooting fish is a barrel" and players will lose interest very quickly.

After that is completed evaluating the player's score at the end of the game and leaderboards will be implemented as will creating the final animation of all the invaders and projectiles.

23 PROJECT PROPOSAL - BLITZ

Project Proposal

Mobile Game – Blitz (Final Name TBD)

Luke Stephens Kehoe, x15018849,
x15018849@student.ncirl.ie

BSc (Hons) in Computing

Gaming and Multimedia

28/09/2015

24 OBJECTIVES & BACKGROUND

The Objective of this project is to create an arena first person shooter that mixes parkour with basketball. It will follow an easy to learn hard to master approach. The idea will be will be two teams each with a goal hoop. Players will fight over a ball which they can hold and try score goals. Players will spawn with the same weapons available to them and there will be other more powerful weapons dotted around the map, which they will fight for control over. The game will be primarily made for pc as that is generally the best platform for first person games.

My main inspiration behind this are games that have come out rather recently Titanfall and Rocket League. Titanfall is a fast paced FPS which incorporated parkour and giant robots. The parkour was by far the standout feature as it allowed the player to transverse the map in seconds and allowed for creative routes.

Rocket league which only came out a couple of months ago and is hugely popular, is effectively 4v4 football with cars. Players ram a giant ball in order to score goals. The beauty of this game is that it is extremely easy to pick up and play but hard to master.

25 TECHNICAL APPROACH

The game itself will be built in Unreal engine. Although I was considering using Valve's source 2 engine, I chose Unreal because of it superior graphical capabilities and proven FPS capabilities with games like unreal tournament

The engine has an inbuilt template for FPS development as well and there are many online tutorials on YouTube and Epic's website for support along with very active forums and subreddit.

Luke Stephens Kehoe

29/09/15

26 LEARNING JOURNALS

26.1 MY REFLECTION - SEPTEMBER

I felt that while doing the research and tutorials with unreal engine that my idea of a FPS parkour basketball game may be too ambitious for me. Instead of that I will go forward with slingshot mobile game and expand upon it, having different game modes, levels and objectives within it. I will be meeting with my supervisor soon and I will try get as much feedback as possible and adjust my project properly started.

26.1.1 INTENDED CHANGES

Next month, I will try to get the project properly started and have the minimal viable product up and running for the midpoint presentation in December. Hopefully by the end of December I will have finished many of the features.

26.2 MY REFLECTION – OCTOBER

26.2.1 SUPERVISOR MEETINGS

Date of Meeting: 4/11/15

Items discussed: Project Idea, Project Plan and Project Technical Approach

This was the first time that I had met my supervisor. We discussed both project ideas that I had proposed. We also talked about how I should approach the technical spec and how to properly plan out the project. As I needed to make a decision on which project to purse quickly, he suggested that I research the marketplace for both ideas.

I researched the marketplace (steam, app store) for variations of both my ideas. The parkour game I had in mind had very little competitors, however any game that was like it was AAA game, with hundreds/thousands of developers. There was surprisingly few mobile

games like the one I had in mind, and the existing ones were sub-par at best. I have decided on the mobile game as I think I can improve upon what is currently out there. Parkour game is outside the scope of this project.

Additionally I have set-up my project so that I can test the game on my android phone and I have found many useful tutorials on YouTube for Unreal engine and the Artificial Intelligence that I will need.

26.2.2 INTENDED CHANGES

Although I recognise that I am slightly behind schedule already I'm fully confident that I can catch up in the next two weeks. I am working on my technical spec now and once that is completed I shall dive into the tutorials.

26.3 MY REFLECTION – NOVEMBER

After using the Oculus Rift (a Virtual Reality headset) in a Computer Graphics lab early into the month I was extremely impressed. I have and have decided to revert back to my original idea of a first person parkour, but develop the entire game with VR in mind. This may be risky but am far more motivated and interested in the project now.

26.3.1 INTENDED CHANGES

I have already dived into creating and refining the movement of the player within the game as this is the most important aspect. When that is competed within I shall start designing the tutorial level.

26.4 MY REFLECTION - DECEMBER

I have not had much time to work on my project because of the upcoming exams and other projects. I have worked on the movement of the character but to very little success. Working without an oculus rift is making this very problematic.

26.4.1 INTENDED CHANGES

After the exams I will continue working on the movement and adjustment of the camera which is the source of all my issues.

26.5 MY REFLECTION MONTH: JANUARY

I have switched back to my original project of the slingshot game. The virtual reality project provided too difficult to accomplish. I have decided on a theme for the slingshot. The enemies coming down from the sky will be birds that are trying to get the worms on the ground. The worms will be defending themselves by launching up cats as projectiles. The game will be then called catapult.

I have gotten a very basic level completed as a prototype. The touch controls are not as well documented as I thought they were going to be. The problem that I am having right now is that I am building a 2D game within a 3D world and I can't have the user controlling the z-axis. This has given me a good idea for a new game mechanic however. The user could be allowed to switch the axis which they are firing from.

26.5.1 INTENDED CHANGES

I am confident that I will be able to fix this issue quite quickly. I will then start working on the AI for the birds and the different types of cats.

26.6 MY REFLECTION MONTH: FEBRUARY

I have created the environment that my game will be played in using a texture pack from the unreal marketplace. I have started working on the AI for my birds. I have ran into a problem where the birds are unable to fly because of a Navigation bounds not being recognised in the air. This is big problem that I am confident that I can resolve. I have found a way around this by laying the game components onto the ground.

26.6.1 INTENDED CHANGES

I intend to find a way around this problem. I still haven't solved the issue that I was having with the spawning of the cats

26.7 MY REFLECTION MONTH: MARCH

I spent most of the month trying to fix the issues I was having with projectile spawning in the wrong place. The project is going quite poorly overall, forcing the game into a 2D environment that is lying down is causing more problems than has solved. The game environment is completely not viewable because it isn't possible to rotate the landscape into the cameras new position

26.7.1 INTENDED CHANGES

I tend on trying to fix these issues for one more week and if I haven't made any progress by then I will take what scripts I have and make a new game concept.

I have changed my project once again. I have used the scripts that I have already made to create a new game where the player has to avoid enemies and gain score. Creating the game is a lot smoother this time then the last two as I have a decent amount of knowledge of how unreal works.

26.8 MY REFLECTION MONTH: APRIL

I have changed my project once again. I have used the scripts that I have already made to create a new game where the player has to avoid enemies and gain score. Creating the game is a lot smoother this time then the last two as I have a decent amount of knowledge of how unreal works.