Project Proposal

**Designing a simple video game with Artificial Intelligence**

**MSc Computer Science**

**Department of Computer Science and Information Systems**

**Birkbeck, University of London**

**Sylwester Stremlau**

**Sstrem01**

**Disclaimer**

|  |  |
| --- | --- |
| I have read and understood the sections of plagiarism in the School Handbook and confirm that the work is my own, with the work of others clearly acknowledged. I give my permission to submit my submission to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software. | |
|  | SYLWESTER STREMLAU |

**Abstract**

This project will focus on creation of a basic 2D video game using the Unity Game Engine and will include sophisticated and responsive Artificial Intelligence that will give players the illusion that they are playing with a real human being. The game will play like early arcade games with simple 2D graphic and basic, easy to understand, objectives, which is a fairly simple game given computer gaming standards, but this is because the work will focus on the Artificial Intelligence. It is planned that it will include an adaptive and responsive game playing strategy, based on Machine Learning techniques. The game will gather data about the player’s behaviour and use it as a training data to fine-tune the Artificial Intelligence to change the difficulty of the game when needed. Artificial Intelligence will use decision and behaviour trees to make decisions with rule base technique elements. The report will describe the process of planning, designing and implementing the game, this includes any design decisions, analysis, background information and evaluation of the results.

Glossary

**Artificial Intelligence (AI)**

Artificial Intelligence is the computer simulation of human behaviour and intelligence. It includes various processes like learning, adapting and thinking to act like a real human. There are various types of AI, from weak (also known as narrow) designed to do a certain task, to more complex, like strong AI that acts and thinks more like a real human being. (Rouse, 2019)

**Rule Based Artificial Intelligence**

Rule based AI is a way to program AI to make decisions based on rules created and modified by a human expert.

**Machine Learning (ML)**

Machine Learning is an implementation of Artificial Intelligence to learn and train itself using data. Machine Learning usually needs training data fed and supervised by an engineer, to look for patterns and similarities so it can make more complex and informed decisions on new data. (Nagy, 2018)

**Video game engine**

A video game engine is a software environment that provides functionality to allow creating and developing games more easily. It provides stuff like rendering engine, physics engine, sound, scripting, animation and other essential functions needed to make a game. (Ward, 2019)

**Game Loop/Gameplay**

Game loop is a very basic loop that controls the flow of the game. A typical game loop is: process inputs, update game and render objects on the screen. (Bethke, 2003)

**Game physics**

Game physics are a simulation of physics that follow more or less how objects behave in real life. Games do not have to follow real life physics but they usually have their own rules that are consistent throughout the game. (Bethke, 2003)

**Graphical User Interface (GUI)**

Graphical user interface is a way for user to interact with an application program through graphical items and indicators. (Bethke, 2003)

Contents

[Chapter 1 - Introduction 1](#_Toc6224218)

[Chapter 2 – Background 3](#_Toc6224219)

[Game development approach 3](#_Toc6224220)

[Artificial Intelligence in Video Game industry 5](#_Toc6224221)

[Utilizing a Game Engine 6](#_Toc6224222)

[Decision Trees 7](#_Toc6224223)

[Behavioral Trees 7](#_Toc6224224)

[Unity overview 8](#_Toc6224225)

[Chapter 3 – Analysis, Requirements and Design 11](#_Toc6224226)

[The Game 11](#_Toc6224227)

[The view 12](#_Toc6224228)

[Controls 13](#_Toc6224229)

[Sprites 13](#_Toc6224230)

[Game engine 14](#_Toc6224231)

[Artificial Intelligence 15](#_Toc6224232)

[Chapter 4 – Experimentation and Evaluation 17](#_Toc6224233)

[Testing 17](#_Toc6224234)

[Success 17](#_Toc6224235)

[Chapter 5 - Planning 20](#_Toc6224236)

[References 22](#_Toc6224237)

# Chapter 1 - Introduction

State the problem you are trying to solve

Why is it worth tackling?

What approaches are available (briefly)?

What approaches have you chosen?

Any special knowledge you presume of the reader

Any special typography or terms

A “road map” of the report . . .

# Chapter 2 – Background

Any information the reader requires in terms of techniques/ technology that isn’t part of the programme you have studies.

Methodology

Research philosophy

# Chapter 3 – Analysis, Requirements and Design

What it says on the title

The play area

Before explaining the game loop and the goal of the game, it makes most sense to first explain the game area. The game area is a classic football pitch that is a rectangle with a line in the middle separating it into two; each side ‘belongs’ to each team. It ‘belongs’ to a team in a sense that the specific team will only spawn on its half of the pitch - this part is just like in the football. There are also goals on each side to which the players have to kick the ball into to score a goal. A team gets one point only when the ball hits the back of the goal, so when the ball is only halfway inside the goal, the team will not get the point. [PICTURE]

At the beginning of the match the ball spawns exactly in the middle of the pitch and whenever it gets hit into the goal, it resets there. This gives each team a fair position from which they can start the game. Player can push the ball with their characters, use swords to hit the ball the travel much faster or use guns to hit (push) it from a distance. The players can score points either by killing the opponent team – for each kill they get one point. Players also get points for kicking the ball into the goal as mentioned above.

There is only one player on each team but the game could be expanded to more players, even up to 11 just like in football although that would put a heavy load on the computer. This size of the team has been chosen because of the complexity of Artificial Intelligence on which this project is focusing on. Bigger size of the teams would make the AI very complex because it would have to dynamically adapt to a lot more situations. For example, in a one-on-one game the AI only needs to follow the ball, block its own goal and try to score the goal. In a two-on-two game the AI would have to do all that plus it would also have to make sure it does not get in the way of its own team mates. More team mates would also mean that in a single player mode (which is when one actual human plays the game) new AI would have to be created which would be in the team with the human player. This means that the AI would have to behave and react to the players’ behaviour differently since a real human is much less predictable. Therefore, focus as of now is on a one-to-one game.

Players have few things that they need to keep track of, Health Points (HP), Stamina Points (SP), Shield Size (SS) and Ammo. The Health Points work very similar as in other games, the player starts with 100HP (which is also the maximum they can have), if it goes below that, they can pick up boxes which replenish their Health Points by a certain number. The amount of HP that each box replenishes is a random number between 10 and 50. When the player HP goes to zero, his position is reset to its default position that is bound to the game character, and one point is given to the opposite team.

To kill the opponents, players have few weapons to use – a sword, a pistol or a machine gun. A sword drains 4 to 6 points of stamina and deals 10 to 15 damage. Stamina points also drains when a player sprints and just like with health points, player starts with 100 and once it goes to zero, player cannot sprint or use sword. Stamina replenishes around 1 point per frame (multiplied by the time.deltatime). Going back to the weapons, players also have guns and each gun has a different speed, ammo, reload time and strength. Pistol for example is much slower and has a smaller magazine size than the Machine Gun but it deals more damage and pushes the ball harder. Machine guns on the other hand have big magazine sizes, that is, they can shoot more bullets before they need to be reloaded, and they are faster and have a shorter reload time. Players can pick up ammo from the random boxes that spawn every 10 to 40 seconds around the map.

The last thing that players have at their disposal is the shield which is a half circle that spawns in front of the character for a few seconds which gets smaller and smaller until it disappears completely. The shield lasts for around 2 seconds and can be used as much as possible. It acts just like a wall that is, whenever a ball hits it, it gets bounced back so it is very useful when defending a goal. Its size allows the player to push the ball away and with little bit of practice can be used to score points. Shield can also be used to block bullets since player gets no damage when the bullet hits the shield. The downside to the shield is that the player cannot use a gun or a sword during the time that the shield is up.

TABLE FOR POINTS

TABLE FOR GUNS

Walls

The goals

Ball

Players

-actions etc.

<https://docs.unity3d.com/ScriptReference/Rigidbody2D.html>

Colliders can be used as triggers which mea

Should include appropriate formal design diagrams/notation as necessary

# Chapter 4 – Implementation

How it works

What is prefab

In Unity there is a very useful physics component to detect collision between objects called Collider2D. It is implemented in a UnityEngine.Physics2DModule and offers a lot of functionality to work with a 2D collision. A Collider has to be attached to a game object and only collides with other objects that have a collider attached to it. There are various types of colliders, Box Collider2D, Circle Collider2D, Capsule Collider2D, Composite Collider2D, Polygon Collider2D and many more that work in 2D or 3D, like Sphere Collider. Colliders have few basic properties like the gameObject, tag, transform, hideFlags and name.

To ‘build’ the walls in the main game area, a number of Box Colliders have been attached to the Football Pitch game object and placed around the pitch. PICTURE. A default setting for a collider is that when another object collides with it, the other object will bounce back if it has a physics component called Rigidbody 2D attached to it which gives the game object control of the physics engine. This means that the game object will be affected by forces controlled from scripts and by gravity. Since this is a top down game, objects should not be affected by the gravity since there is no real up or down but they should be affected by forces. Going back to the colliders, another setting for them is to act like a trigger. This means that whenever another object collides with it, it will ‘pass’ through it but a message will be sent in a Collision2D format that includes all the information about the colliding object. PICTURE. The second type of collider has been added to a bullet prefab and a basic script has been added which destroys the prefab whenever it hits another object. These walls are very important because they also stop the player and the ball from moving further than the allowed game area. Circle colliders had to be added in the corners of the walls because there was a bug where a ball would sometimes stop just next to the wall which would prevent it from bouncing off it to the side, so it would get stuck and only be able to move alongside the wall. Circle colliders helped to overcome this bug in a way that whenever the ball reaches the corner it would collide with the circle which would bounce the ball away to the side. PICTURE.

The football pitch does not have any other components than the sprite which is just a rectangle with lines around it and one line in the middle. However there is another part of the football pitch, the goals. The goals are very similar to the playing field but they are made as separate objects they needed to have different tags so it would be easier to write a method that recognizes which goal the ball has hit and which team should get a point. At first it was planned that the goal would be just a straight line and whenever a ball would collide with it, it would count as a goal. However, after testing it turned out this is not the most effective and fair placement of the goals so new goals had to be designed. The new goal allow the player to get inside it similarly like in the game Rocket League, in that game a goal is only counted when the entire ball enter the goal. This allows the players to defend the goal more easily.

The ball is a more complex object than the wall because there is a bit more information and functionality needed to make it work. First of all, there is a basic Circle Collider2D set up just like in the walls, and the Rigidbody 2D. The most important features here are the material of the object, mass and linear drag. Gravity can be ignored since as mentioned above, the gravity should not affect any of the objects since there is no ‘real’ up or down. The material of the object defines its friction and the bounciness. Since it is a ball, its friction should be very small and bounciness should be high, in this case friction is set to 0.1 (or 10%) and bounciness 1 which is a default setting. These setting allow it to behave just like a real ball. Linear drag has been set up with the mass, the higher the mass the harder it is to push the ball therefore the mass had to be kept relatively small. However the side effect of that was that the ball would travel too fast so a linear drag had to be set up, this required few hours of testing and playing around and the final outcome was 0.4 (40%) linear drag. The numbers here are very ambiguous but the most important parts are that friction, bounciness, mass and linear drag had all been used to fine-tune the way the ball behaves.

Another important part of the ball is the ‘ball controller’ script which keeps track of the goals and updates

Players object are the most complex because there are much more information needed to be passed through it and it has more functionality that is needed to make it work. Just like the walls and the ball it has all the components that take care of the collision and movement. Since the sprites are circular, Circle Collision2D components have been used to make give the best performance possible. Complex collision detection may sometimes be processor heavy which can lead to slow performance and bugs. Rigidbody 2D has also been used but this time it was much more straightforward to set up because player has to move only when the button is pressed or when the player tilts the joystick stick in a certain direction. The player does not stop immediately when a player stops pressing a button but it still stops relatively fast compared to the ball which bounces around. If a slow slow stop time and a high mass were set up it would make the player’s character feel like it is sliding on ice which is not the desired effect. The game is fast paced therefore it makes most sense to make the controls feel very responsive and easy to control. Therefore it took a lot of tests to come up with the best settings for the player movement.

Wall

Behaviour

Describe how the implementation maps onto the design you have already discussed.

You should use “code snippets” to illustrate special features of your work or difficult (awkward) bits of coding. Don’t make any of these snippets longer than half a page (and include line numbers if possible). If the code fragment is longer than half a page then break it up into smaller bits.

Describe the code, both in terms of the overall architecture and in terms of the snippets. Make sure the reader understands what you have done and why!

# Chapter 5 – Experimentation and Evaluation

Again, what it says on the tin!

If you haven’t done too much testing (for instance it is GUI based) then include a “walk-through” of the application with screenshots showing the scenarios in which the application can be used. After all, the examiners may not be near a computer to actually “run” your code.

You also need to clearly show how you have evaluated your work and show how it meets the original aims and objectives.

# Chapter 6 – Conclusions

What have you learnt?

Basically, “reflect” on the work you have done.

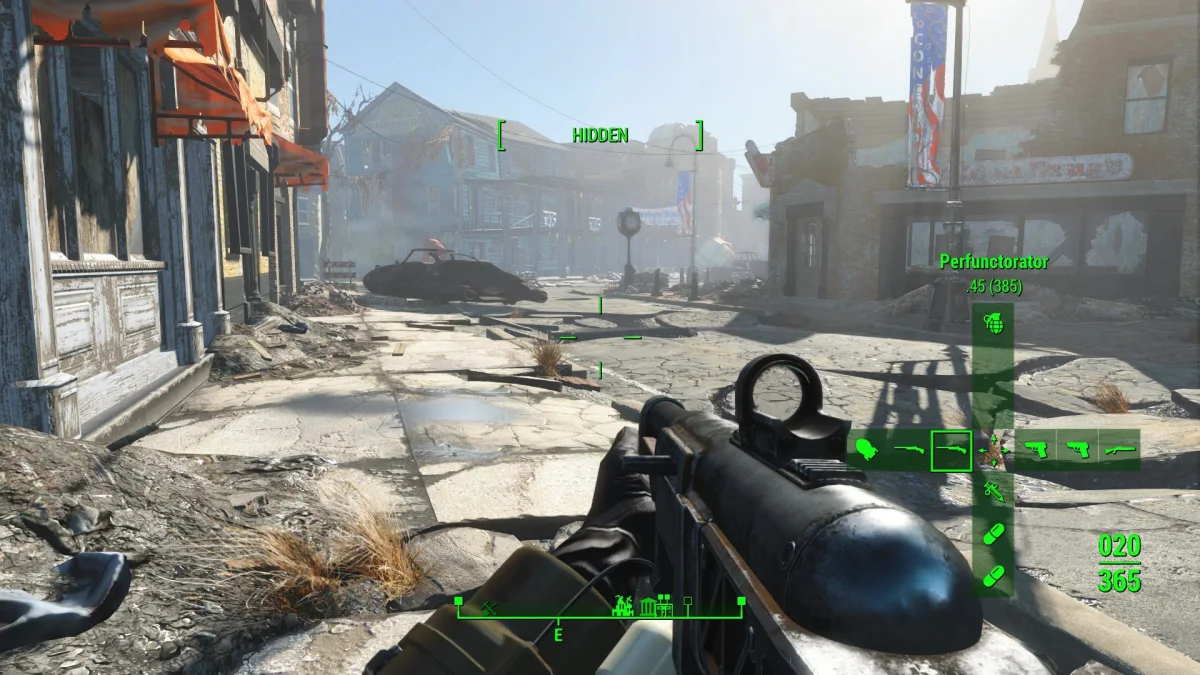
What additional features/extensions can be done to the work and/or what would you have done if you had more time.

# Appendices

Include design diagrams, data formats, etc.

Basically, don’t overfill the report.

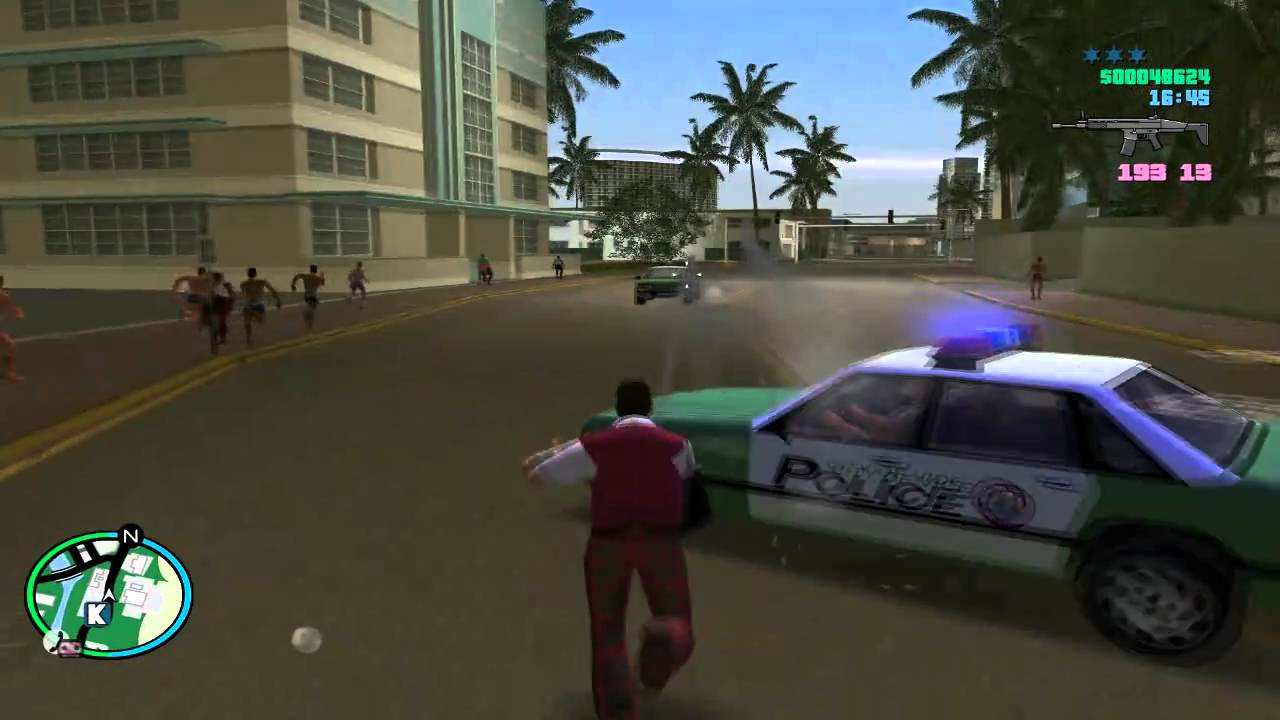
Link to the GitHub repository

Game UI









# References