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

The main aim of this project is to create an enjoyable game experience with computer controlled enemies that is complex enough to be able to challenge a player and make each game exciting and distinct. The main objectives of this project are to create a working game prototype, create a stable and intuitive gameplay, research and design adaptive Artificial Intelligence, research ways other developers improve AI in their games and lastly, use machine learning to improve the AI to be more human-like.

As I only have experience in Java and Python, doing this project will give me an opportunity to learn a completely new environment, which is a Unity engine, and a completely new language which is C#. This will allow me to grow as a programmer, learn new techniques and build my portfolio. Unity engine is the industry standard for gaming, John Riccitiello said "It's pretty much half of all games period", about the amount of games made in Unity (Dillet, 2019). And being able to work with the industry standard tools will inspire, challenge and motivate me and on top of that, it will show my potential employers that I am eager to adapt and learn.

Artificial Intelligence is also a very big and important aspect in, not only just gaming, but computer industry overall. Therefore making this project will allow me to dig deeper into the AI, understand its workings and develop a better appreciation for video games. I am also planning to work with Artificial Intelligence in the future either developing it for games or for research. This project will give me the experience and opportunity to learn about it in more detail and maybe even help me with my first job.

This project will focus on creating a basic video game and designing Artificial Intelligence that will behave and challenge players similar to a real human being. The game will be kept very basic; inspired by games like Rocket League, Grand Theft Auto 2 and FIFA that provide interesting, challenging and arcade-like experience to users.

In this game there will be two teams on a football pitch, one ball and two goals just like in a classic football. Each team will have to work together to score a goal by using various weapons like melee, guns etc. to hit the ball to the opponents goal. Each player will have their own health points and the game will be timed, the team with the highest score wins. This type of game has been chosen because it is very basic, easy to implement and there is a lot of room to design the Artificial Intelligence. In a video game, Artificial Intelligence is a computer controlled opponent that makes decisions based on rules and other data either fed by the developer or gathered throughout the game. In this game, the AI will have to be programmed to work with various team sizes which will have to behave differently in each situation. This is a big advantage because it will be very easy to scale the complexity of the Artificial Intelligence up or down, so the scope of the project will be very flexible which will help to finish it on time.

Therefore this project will not focus so much on refining and polishing the game mechanics because this can take a full team to develop - In an interview with The Guardian, Dave Hagewood, CEO of Video

Game Company Psyonix said that the process of refinement of Rocket League took over seven years, with a team of 15 people. This project will instead focus on designing the game AI to be able to adapt to the difficulty of the player and give an impression that the player is playing against a real human being. As of now, the scope is to make the AI playable in various game modes and team sizes explained in later chapters. For the same reason, the graphics of the game will be kept as simple as possible – basic geometric shapes should be enough.

To keep the scope of the project small, the AI will be rule based and will use decision trees to decide what to do and behaviour trees will be used to control its behaviour. It will be a small game so there is no need to make these trees complicated and the goal will be to make them as efficient as possible by using as least rules as possible. It is also planned that the game will include an adaptive 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 input to fine-tune the Artificial Intelligence so it adapts its difficulty to the player. (Tricentis, 2019; Nagy, 2018)

In Chapter 2 I will go over the background of the game industry, where it is right now and where it is heading. I will also go through one of the best AI developed for video games like F.E.A.R. and The Sims and describe what techniques and strategies the developers of these games used.

In Chapter 3 I will analyse how the game will need to be made, all its aspects and how I will use the knowledge of other games to design my own. I will then discuss the language, game engine and framework that I intend to use to create this game.

In Chapter 4 I will go through the techniques I will use to test the game and how I will determine whether the project was successful and if it meets the aims and objectives.

In Chapter 5 I will present a Grantt chart which says how I plan to use the time most effectively to make sure the project is finished on time and to a highest standard.

CHAPTER 2 – BACKGROUND

This chapter will first explore some of the reasons why AI in video games has stagnated even though the video game industry has been gaining popularity and has been growing rapidly. Then it will look into some of the most popular Artificial Intelligence examples in video games and describe the techniques that the developers used to develop them.

GAME DEVELOPMENT APPROACH

In an article for Gamasutra, G.Lopes and R. Kuhnen propose a layered view of the game design which they explain that "each layer corresponds to a generalization or abstraction of the layers below it" (G. Lopes, R.Kuhnen, 2007). What they propose is the whole process should be broken down into the following layers: Concept, Context, Core (Features and Content), Mechanics and Verbs. Each step can be done sequentially from top-down or bottom up depending on the requirements. These two approaches are very similar to the Top-down/stub and Bottom-up software development techniques. In top-down for example, high level methods and the logic are first designed, dummy methods called stubs are inserted in place of actual methods, the programs is then tested and the whole process is repeated on the lower level.

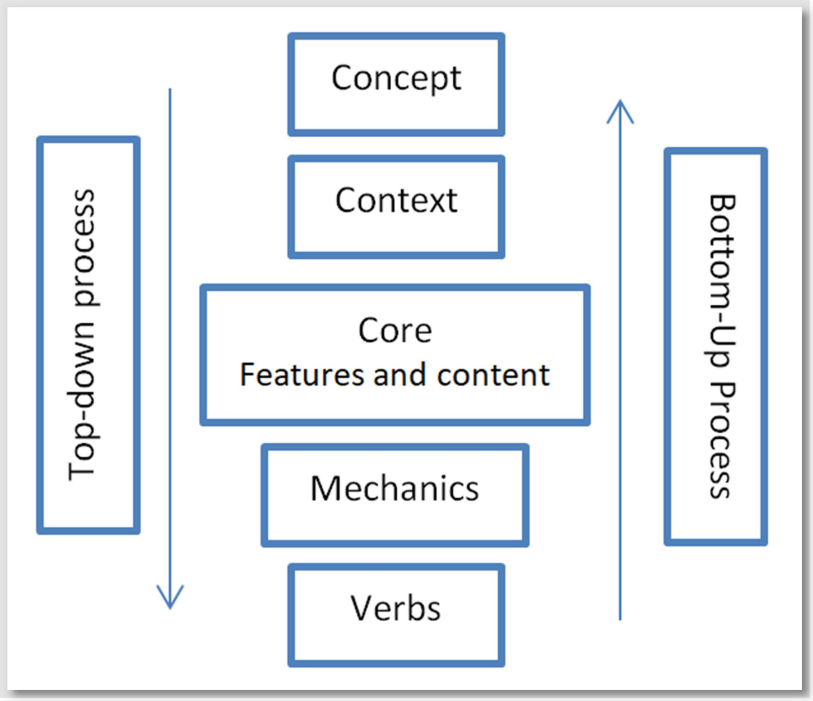


Figure 1. Game development process proposed by G.Lopes and R.Kuhnen

For this project a revised top-down approach proposed by Lopes and Kuhnen mixed with agile and waterfall methodologies will be used because as E. Bethke said in his book “Game Development and Production”, at its core, game development is just a programming project development with few extra steps.

As shown in Figure 2, most of the steps will overlap and will be refined as the project progresses just like in agile methodology. To avoid wasting time, some of these steps like concept and context may get less attention than the other steps like core or mechanics because this project focuses on making AI first and not on making a commercially successful game but a playable one. (Bethke, 2003; Lopes and Kuhnen, 2019)

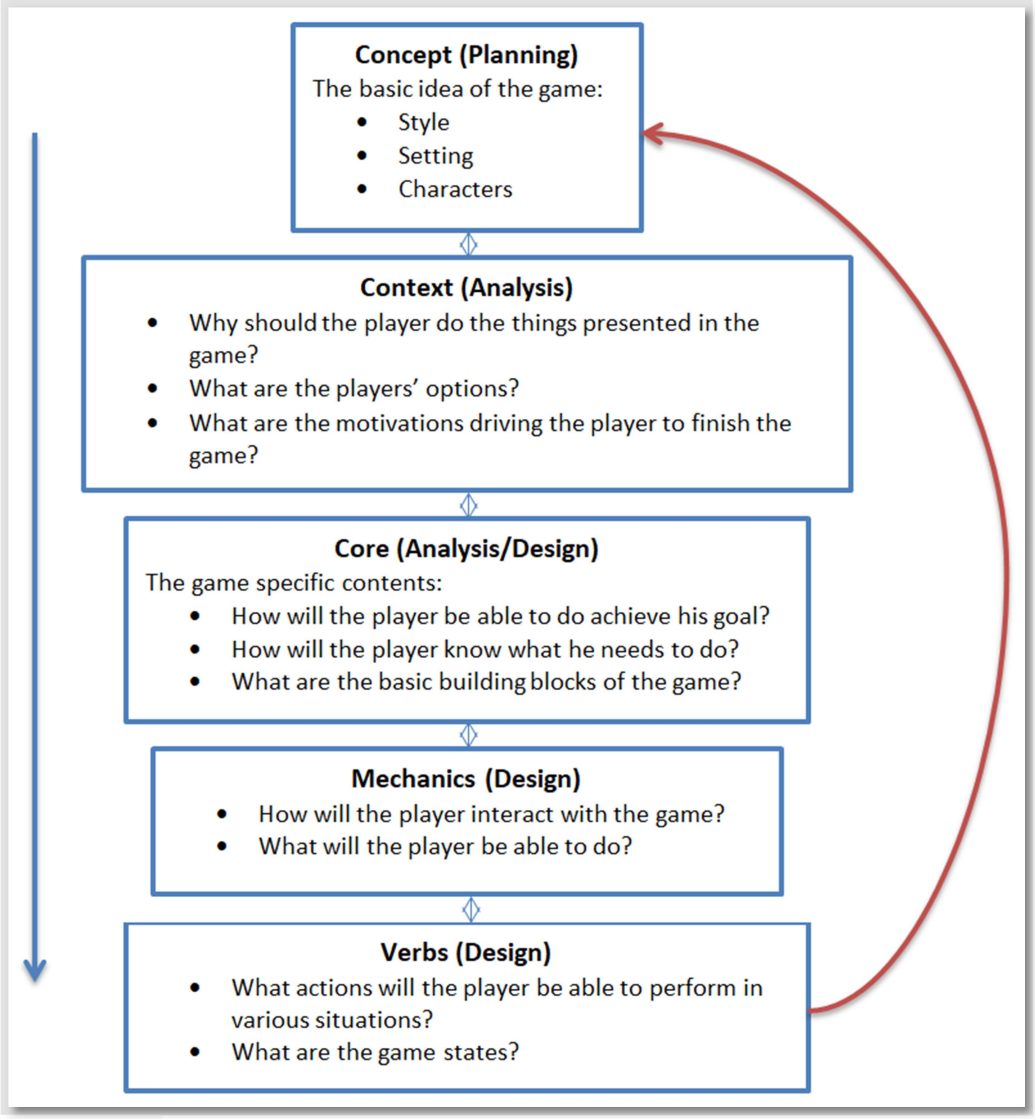


Figure 2. Proposed game development approach for this project.

ARTIFICIAL INTELLIGENCE IN VIDEO GAME INDUSTRY

With the emergence of the new technology which allowed for games to be more immersive and complex, video game industry has blew up so much, it surpassed movie and music industry. According to Global Games Market Revenues by Newzoo, video game industry has generated more money than music and movie industry combined for the past 11 years. In 2021 it is estimated that it will make $180,000m while movie industry will only make around $51,000m and music industry will make $22,000m.

One would think the games are getting more and more sophisticated with all the new technology and budget but it may not entirely true. While in some aspects like graphics, storytelling or audio, they do, in other aspects like Artificial Intelligence there may be stagnation. According to numerous articles, one of the best AI is in the game called F.E.A.R. which was released in 17th October 2005, almost fourteen years ago – around the time that video game industry started really blowing up. (Horti, 2019; Orkin, 2006)

The techniques and systems used in F.E.A.R., while not entirely new or original were implemented extremely well and modern FPS' video game AI is usually not much more complicated or sophisticated. There are various reasons for this problem; the obvious one is the lack of funds thus the focus shifts to different aspects of the game like graphics, audio or online gameplay. Another reason may be the focus on the online gameplay – why develop complex and expensive AI that behaves like humans when players can play against each other (real humans)?

Another excellent example of game AI is the EA's life simulation game The Sims, recognized as one of the most influential AI of all time (Bourse, 2012). In this game player controls a family (called Sims), designs their house, gives them orders and does other numerous tasks. Sims can make decisions and survive on their own, however their behaviour is deliberately limited so players have a reason to play and feel like their decisions matter, otherwise the game would just play itself.

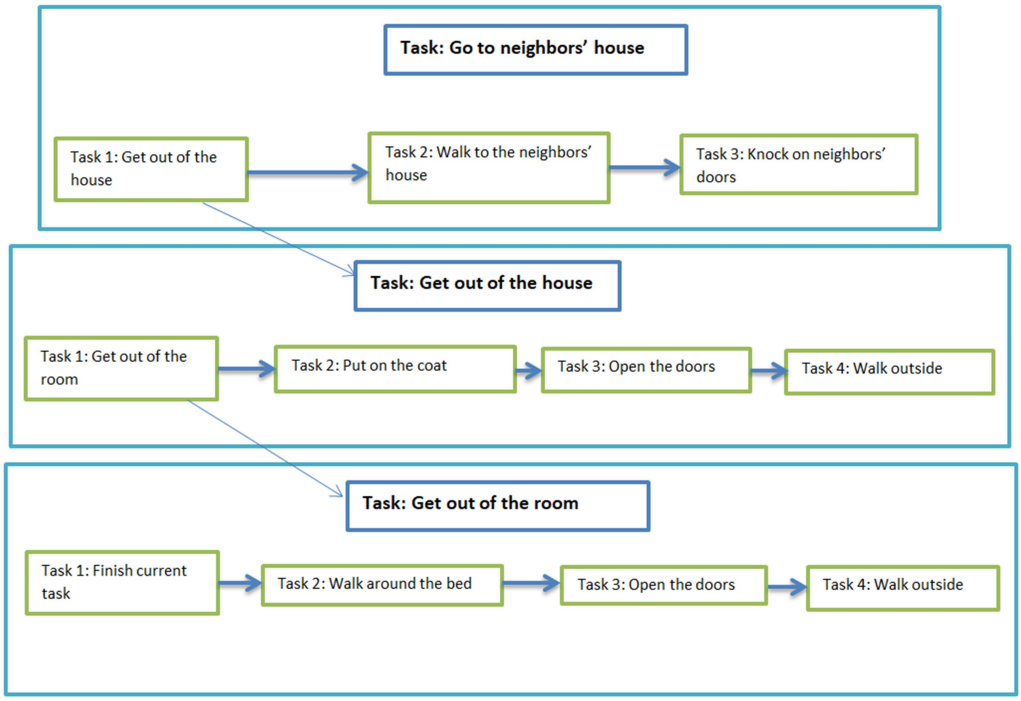


Figure 3. The Sims Hierarchal Pathfinding AI example.

The Sims use Hierarchal Pathfinding which means that for example, when the Sim decides to go to the neighbour, this step is broken down into smaller and smaller pieces, first he needs to go from his house to the neighbour's house, this means that first he needs to get out from his house, which means he needs to get out of the room that he's in and put on a coat, to put on a coat he needs to take the coat out and so on (Boarse, 2012). This is mixed with smoothing algorithms to offer a very smooth path and make the Sims more realistic. HP is a great way to make AI feel realistic and intelligent, can be applied in a fairly easy way and is a great way to program AI to make seemingly complex decisions by breaking them down. (Botea, Muller and Schaffer, 2019)

Utilizing a Game Engine

When starting a game, at one point in the design process developers have to decide if they need to create their own engine or if they will be using a one that is available on the market at the moment. This decision depends on various variables like the budget, requirements, style of the game, resources, among others. This decision is very important because the game engine is the core building block upon which the game is built on. Game engine gives developers tools like graphics (3D modeling, 2D/3D rendering, animation), logic (physics, collision detection, artificial intelligence), audio (sound, music), and many others. Usually, all of these aspects are broken down and a different team works on each of them, which are then brought together inside the game engine. For example, one team would work on the animation aspect and another team would work on the game physics. To make the best use of time,

each team would have to work independently of each other therefore once each team has finished, game engine would allow all of these aspects to be brought together in one place. (Pitts, 2014; cdprojekt.com, 2019; BBC News, 2018; Campbell, 2018)

Decision Trees

Decision tree is a basic tool to help represent decisions in a tree-like model. It has a root and child nodes used to make a decision. Video game AI uses decision trees to make various decisions like shoot, walk, run, hide etc. and these decisions can be either very simple or very complex. Figure 4 shows a basic example how decision tree would be used in a basic video game AI. (Bethke, 2013)

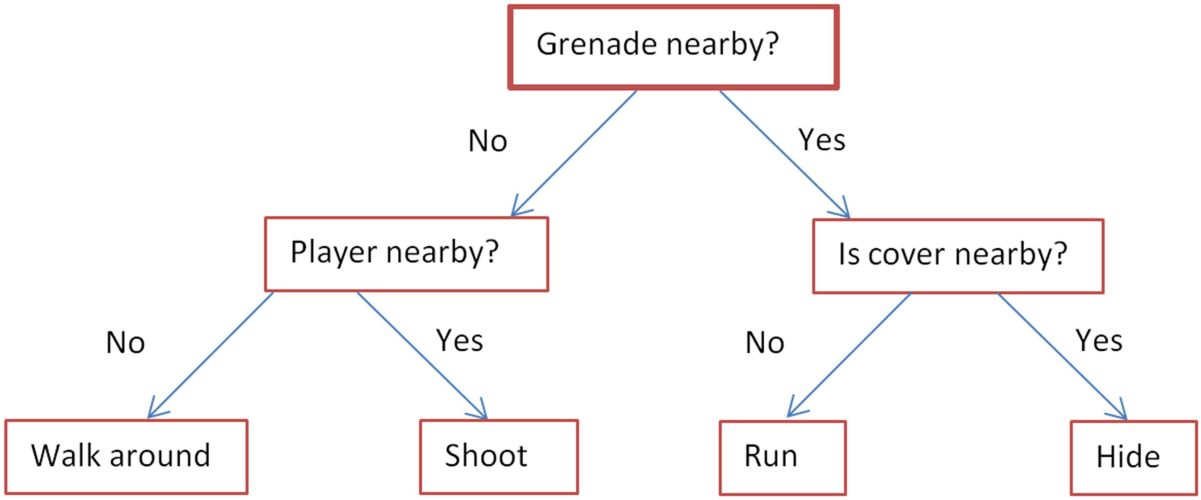


Figure 4. Example of a basic Artificial Intelligence decision tree

Behavioral Trees

A behavior tree is very similar to a decision tree but a bit more complicated which allows for a more sophisticated behavior. Figure 5 shows a basic behavior tree from a Unity game kit that will be used in this project. A tree is traversed with each game update and one of three states is returned for each node. These states are: Success – it has finished its task, Failure – it has failed and, Continue – it has not finished its task yet. (Unity, 2017)

Selector executes each node until one of them returns success then exits without executing the rest. The test node only executes its child if the test returns true, this is similar to the decision tree in Figure 4, the "Is cover nearby" is only executed if "Is grenade nearby" is true. Tests can vary depending on the situations, and can fail or finish depending on various variables. For example, it can be tested if the AI has low health, if it returns true, the Function would be executed which could be another tree which checks if there is any health around, if yes, the next function would be executed that makes the enemy get the health and so on. If on the other hand, if enemy does not have low health, this test would fail but another test may check if enemy's health is high and would execute another function based on that.

Therefore it is important to think of all possible outcomes so the enemy knows how to behave according to the situation.

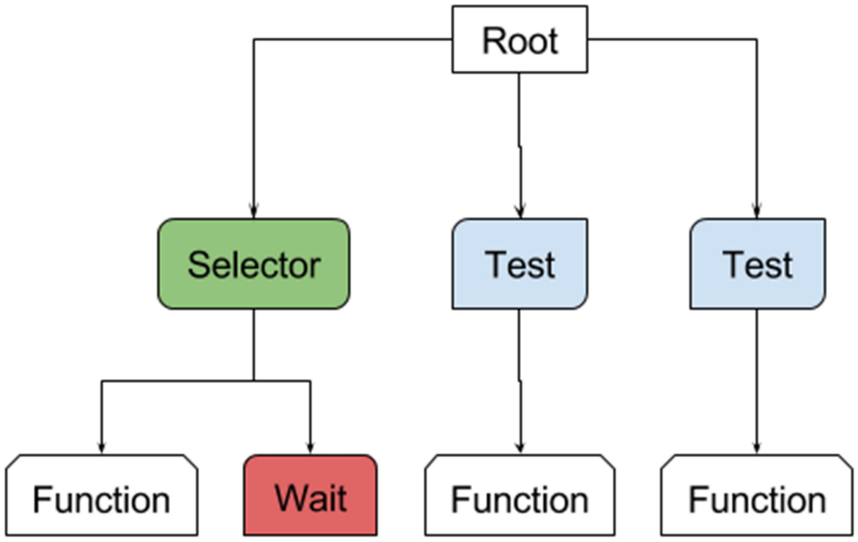


Figure 5. Behavior tree from Unity game toolkit. (Unity 2017)

UNITY OVERVIEW

Unity game engine provides all of the tools discussed above and many more. It provides 2D and 3D graphics which will be important to finish and improve this project. It provides tools to create, design and implement artificial intelligence into the game with ease. It also provides templates that users can learn from and use to help design their game.

Unity allows to playtest the game whenever a user wants so it is very easy to test out any new function when it is implemented. Every asset in the game is sorted in an asset window where users can select and edit everything very easily using an inspector window as seen in Figure 7.

It also provides a very easy and useful way to compile and edit the code in an object oriented programming paradigm as it can be seen in Figure 8.

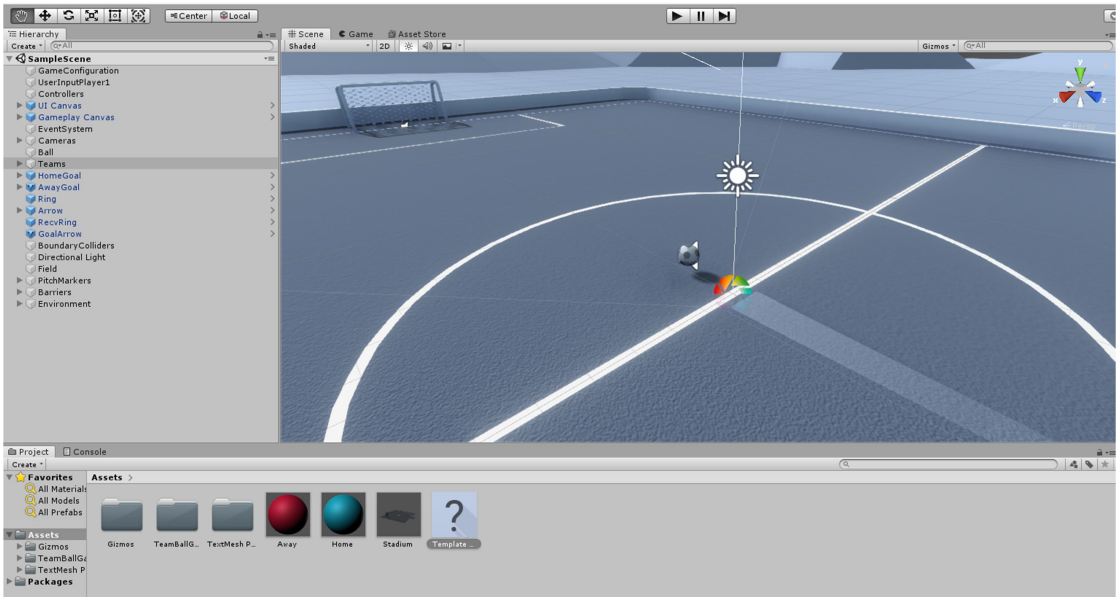


Figure 6. A basic window of unity

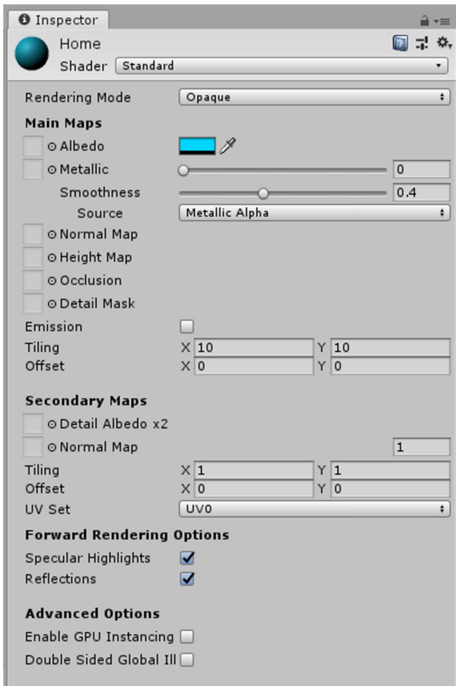


Figure 7. Unity Inspector window

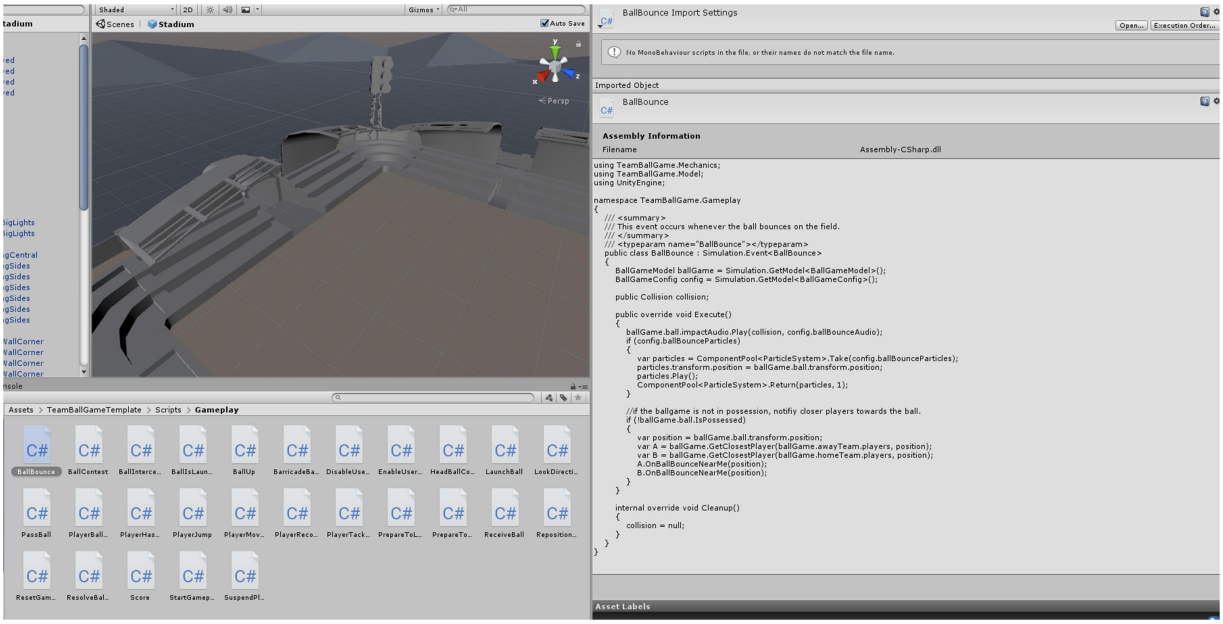


Figure 8. Unity programming interface

As it can be seen in Figure 8, Unity also provides a 3D modelling and map editor where users can create their own models and map very easily without having to use third party software.

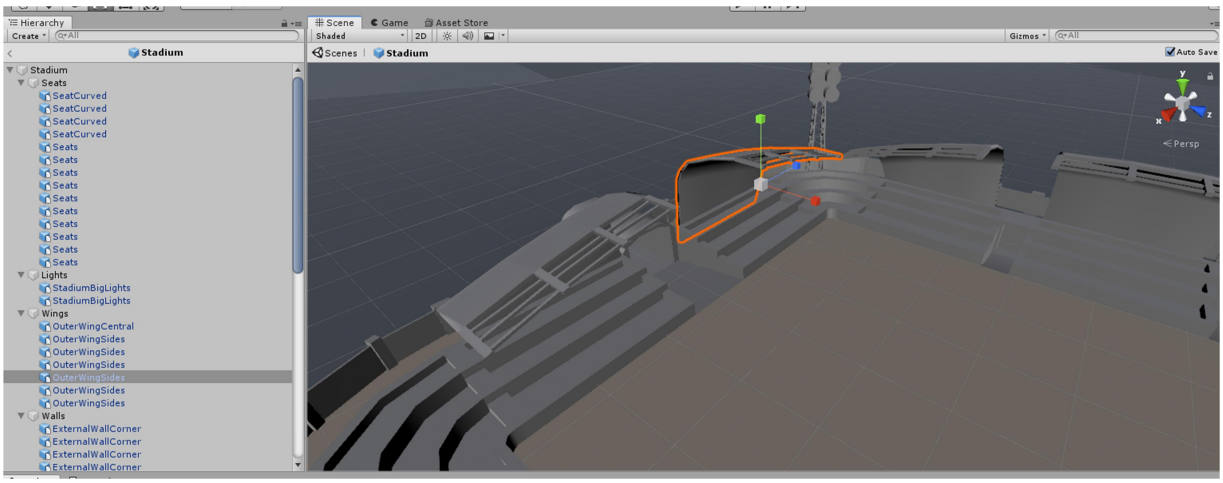


Figure 9. Unity map editor.

Therefore Unity is a great fit for this project because it will allow creating a complex game inside one place and provides all the necessary tools to do so.

CHAPTER 3 – ANALYSIS, REQUIREMENTS AND DESIGN

This chapter will go over the game design and basic plans for this project. It will explain how the game will look, how everything will be brought together and how the game will play. It will also explain how the game AI will be designed and implemented, and what approach will be used.

At the start of the project, waterfall methodology will be used because it uses a linear sequential approach and planning, designing and analysing will be very important at the start for the project to succeed. However, the outcome of these phases will be used just as a guideline and once the development will begin, agile methodology will be used. Agile will be very useful because requirements and the scope of the project may need to slightly change as the project progresses and it will also help to avoid making it too small or too ambitious that will be impossible to finish. (Edeki, 2015)

THE GAME

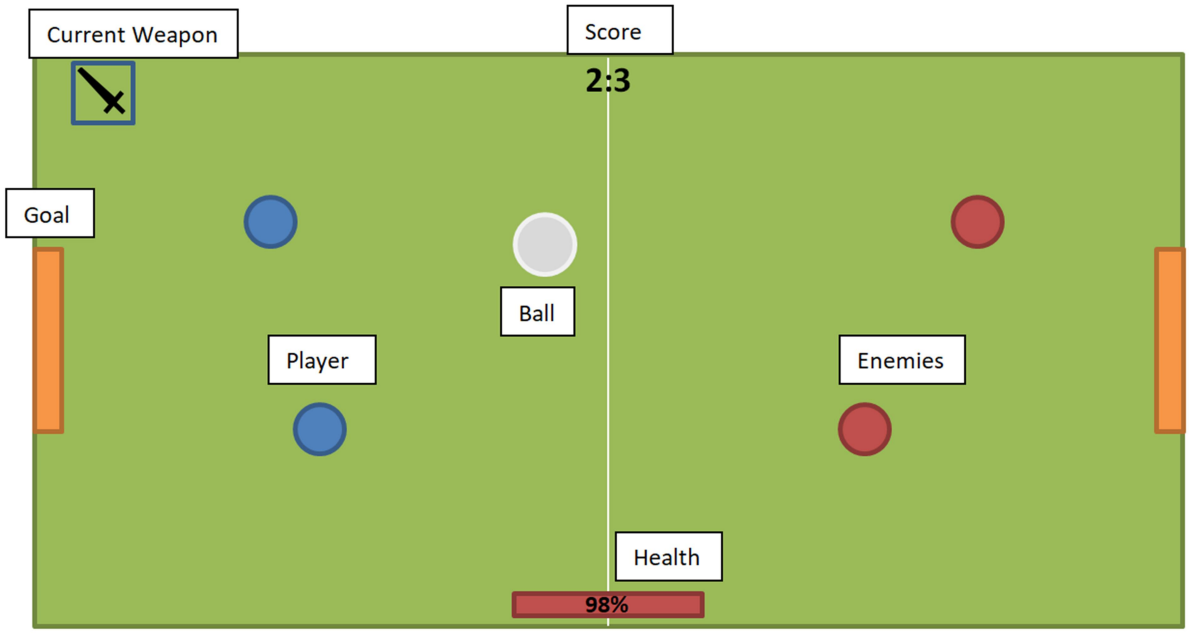


Figure 10. Basic design of the game project.

The game will be a top down shooter/fighting/football game similar to Rocket League, there will be two goals, one ball and at least one player on each team. Each player will try to shoot the ball into the opponent's goal to earn a point. Team with the highest amount of points at the end of the game, wins. Since there are already successful football games like EA's FIFA and PES' Pro Evolution Soccer, this game will be slightly different and will not exactly follow the classic football rules. For example, there will be no offside; the ball will bounce off the sides and there will be no penalties etc. (TheFa.com, 2019)

To make the game more interesting and unique, players will have health points, stamina points and weapons like swords or guns – just like in a game Hotline Miami – with which they will be able to attack other players to gain additional points, score and to for example shoot them as they are about to kick the ball in to the goal. The default positions of the players should be researched during the testing to make sure that the flow of the game is not broken.

It is considered that the ball should also have health points, although much higher than the players to avoid breaking the flow of the game. This would create unique situations and add tension to the game. This would also add variation because players would have to ask themselves if they want to use swords that can send the ball really far but are not able to hit the ball from a far or if they want to use guns that are really good from the far but are not as good in a close range.

THE VIEW

Figure 11. GTA 1 video game 1997

The 'camera' will be pointed from above like in games like Grand Theft Auto 1, Hotline Miami, Overcooked, Undertale and many others. While this Point of View (POV) may seem out-dated, the gaming community and market for this type of games is still very big. For example, Undertale, released in September 2015, was one of the best-selling games on the digital game market Steam, with 530,343 copies sold by the end of 201 5(Galyonkin, 2019). Undertale uses top down view and retro graphics in its gameplay and it does not stop people from buying and enjoying it. The aim will be to make the game look very intuitive so anyone can start a game and after just few minutes, be able to understand and see what is going on just like in the game shown in Figure 11, player shown only most important aspects of the game. In this game the objectives are easy to see at the bottom of the screen, his position in the game world map is displayed at the top and stuff like health, money etc. are shown in the Head-up Display so player knows exactly what is going on at all times.

CONTROLS

The game will be focused on gamepad controls so the players will be able to control the character in 360 degrees. Player will tilt the analogue stick in the direction they want to move and the character will follow their lead. Alternative controls for keyboard and mouse controls would be to use arrow keys (or WASD) for walking and mouse for pointing in the direction to move. This should be researched, tested and compared with other games with the goal to make the controls intuitive, simple and easy to use.

SPRITES

Sprite is a 2D object used for players, enemies, props and various other 2D elements in a video game. In Unity, SpriteRenderer is used to render the sprite graphics obtained from the bitmap images on the screen. (Unity, 2018)

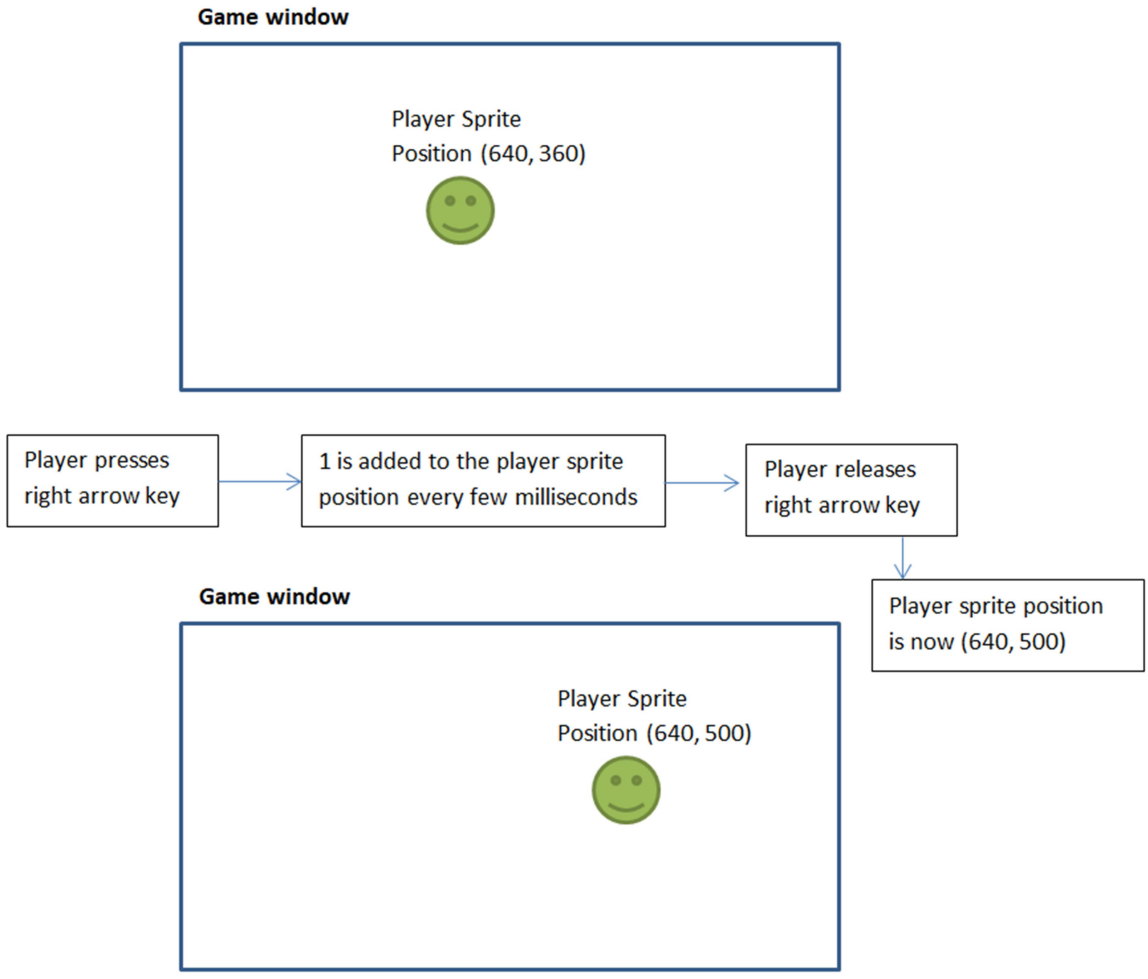


Figure 12. Player movement

As it can be seen in the Figure 12, when a player hits an arrow key, a function is triggered that basically just adds a certain amount to the sprites' x or y position. A player usually has control of one sprite called

the player sprite, although it is not much different than the other sprites. Player's movement can be more sophisticated by using various algorithms to calculate acceleration and velocity to make the player movement smoother. Playing with these algorithms can create various effects, for example, in New Super Mario Bros the player acceleration and velocity is fixed for most of the game but in ice levels, it is much harder to control Mario because he keeps sliding. This effect could be achieved by making the acceleration time much slower thus giving an effect that the character is sliding because it takes him few seconds to rich full speed. (Nintendo, 2006)

The game window is usually fixed and is used to help calculate the position of the player. However, if the map of the game is much larger than the game window, player would control the camera instead of the player – that is, the player position would be kept in the middle and the position on the screen of everything else would be updated which would give an impression that the player is walking. The same algorithms like described above can still be applied to give a smooth experience.

GAME ENGINE

There are various approaches available to create games and to write the Artificial Intelligence. Some of them are more popular and robust and others are smaller but still very popular. Three choices that fit the projects requirements have been taken into consideration these are Unity, GameMaker and PyGame.

Unity

The first choice that may be the most popular is the Unity engine, a cross platform game engine that is used in half of the world's games, developed by Unity Technologies (Unity, 2018). Unity uses scripting API in C# programming language and provides users with tool to develop 2D and 3D games. It supports various APIs like Direct3D, OpenGL, WebGL and support development for Android, Windows, PlayStation and many others.

GameMaker

GameMaker Studio Studio developed by YoYo games is a video game engine supporten on platforms like Windows, MacOS and Ubuntu among others. It has a very basic Drag and Drop (DnD) scripting tool that can be used by novice creators but is complex enough to make complex and sophisticated games. Some notable games developed in this game engine are Undertale, Hotline Miami and Nidhogg.

Pygame

PyGame is open-source library written for the Python programming language. It provides support mainly for Windows, Linux, MacOS but it does not provide support for the consoles. It is not a very complicated

or popular tool (anecdotally it is usually used as a learning tool) but it can be very effective for small applications.

For this project, Unity engine seems like the best fit because it is industry standard so it has numerous options like 2D game support, lots of tutorial material, is very popular and is widely supported. It also has a very easy interface which will allow finishing the main gameplay in a timely manner so more time can be spent perfecting the AI. Unity is also free which means that no additional costs will be needed. The version of Unity that will be used is 2018.3. Unity is an excellent tool to use because it has all necessary tools and options described above and is easy to use. (Unity, 2018)

Unity uses API written in C# which is a multi-paradigm programming language developed by Microsoft in 2000s, with current version being 7.3. This language has been chosen because it is the main Unity scripting language, is similar to Java in which I have experience therefore it will be much easier to start working on and understanding than any other language. (Microsoft, 2018)

Artificial Intelligence

Artificial Intelligence will be written in C# as this is one of the languages that Unity supports. One of the reasons that C# will be used instead of other language is that most tutorials, assets and examples use C# so even without previous knowledge it will be fairly easy to get into. Another advantage is that C# is very similar to Java because they are both object oriented programming languages and have a similar syntax, since I have experience in Java, I should not have issues understanding and using C# without having to go through lots of tutorials.

The AI will use decision trees based on rules to make their move and Hierarchal Pathfinding as explained in Chapter 2. An example of a rule could be to check if a new power up is available, the check if a power up is already equipped and behaving accordingly. These will have to be kept simple as AI will have to make decisions very quickly and efficiently in real time. Although a lag or delay to the decisions may be added deliberately to create various difficulties to fit various types of players. The best reaction time for the AI should be researched and planned during the design process of the AI and during the Quality Assurance step.

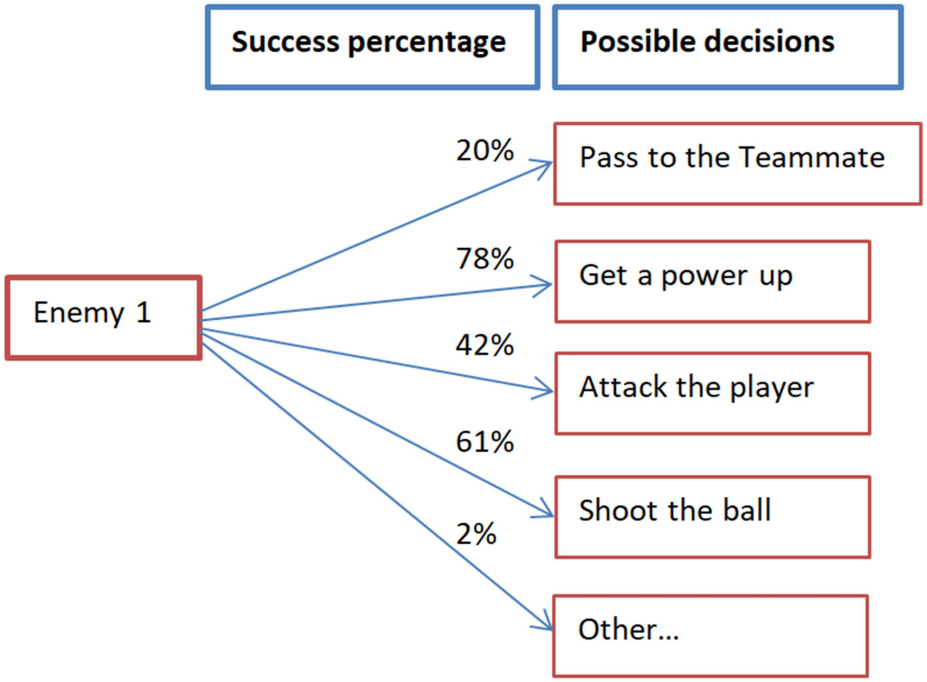


Figure 13. Example decision tree.

The design and programming of the AI will have to be different for different team sizes, for example, AI would have to play in team in a 2vs2 game and on their own in a 1vs1 game. This will mean that their decision trees will have to be different because a team aspect will have to be added. Most of the time the AI's goal will be to score a goal but sometimes that may change, for example a new power up will be available, and so their new short term goal would be to get the power up but they may be on the other side of the pitch so first they would have to kick the ball to the other side of the pitch then fight the enemy and sprint to the power up, this would be when the Hierarchal Pathfinding would come in. Figure 6 shows a very basic example of how it would make its decisions in real time. This would most likely be much more complicated since it would have smoothing algorithms and other calculations on top. The best way that enemy AI should move should be researched and planned in the planning and design parts of the project.

CHAPTER 4 – EXPERIMENTATION AND EVALUATION

This chapter will go over the ways that will be used to test the game and then it will describe some ways that will be used to judge the projects failure.

TESTING

As in agile methodologies, testing is done after each iteration, which is after each function has been added. Since this project will be done using agile methodologies, each function will be tested for bugs and unforeseen behaviour after it is implemented. This is important because the project can get very big and this will ensure that it works as well as it should at all times. (Edeki, 2015)

An indication that a project or a game has failed is when a lot of the code does not seem to work together. This can lead to the game becoming unplayable, not enjoyable or even frustrating for the players, which can make them stop playing the game and lose trust in the developers. An example of that kind of game is No Man's Sky which was released in such a bad state that it took few years for the game company to gain the trust of the gamers back, but by then their audience got really small and no one was interested to play the game. Therefore if the game will have lots of faulty code and players will not be able to enjoy the game, it will mean that the project was not able to meet one part of its aim which is to create an enjoyable game experience. This will be decided by going through the game reports and reviews from the users who played the game and deciding what amount of people enjoyed the game and what amount of people did not, or only partially enjoyed and comparing it to other games.

SUCCESS

A first hint that the project has been successful will be that the game is playable and enjoyable. That will mean that the first part of the main aim which is to create an enjoyable game experience has been met. That will first involve playing the game with friends instead of enemy AI, over and over again to find bugs, imbalances or any other abnormalities. A report about how the game plays, how fun it is, how it performs and how all components work together should be written at the end of this that will detail all of these aspects.

Next the same will be done but with enemy AI by playing on various difficulties, looking for any issues, checking how the AI behaves, how are the functions working together, and looking for any situation where AI is not able to decide what to do. The output of that will again be a report judging and comparing the game to other indie and professional games and trying to decide if it could compete with them.

It is very hard to judge a game by one player since fun can be an individual thing, therefore once the reports have been written, the game will be sent out to other players willing to test the game. They will

be sent a free copy with a condition that they fill out a basic survey with various questions asking them to judge the game and their experience. This should be a great way to truly see if the project has been successful because it would be a feedback from the potential buyers of the game and from players of various skill and background. One drawback of this is that some players play very casually and do not think of the games critically.

A/B testing could be used to achieve the best results. A/B testing, also known as split testing, is a method of comparison of two versions of an app or a game against each other to figure out which one is better (King, Churchill and Tan, 2017). For this project A/B testing would be great because two variants need to be tested; the user experience and the performance of the game. Therefore the test group will be split into two, first group would be users that are casual gamers and/or do not have much experience with games, and the second group would be users that are more interested in games and the technical aspects of the games. The first group would be shown questions about the fun of the game, flow and overall user experience. Both groups will include similar users which will most likely be a group of people in 12-25 years of age group that have at least some basic familiarity with gaming.

Table 1 shows example questions that will appear on the survey although there will be much more questions that will be carefully planned and designed to maximise the results. The table 2 shows example questions that will appear on the second survey which ask about the quality of the game, mechanics, bugs and other technical aspects. A way to split the group and choose which users are shown which survey needs to be planned and decided during the project planning phase. Appropriate statistical analysis will need to be researched and designed to make sure they are fairly compared.

Table 1. An example of the first survey - User experience

Question	**Comments**	**Agree**	**Neutral**	**Disagree**
Overall the game was fun				
The game experience was smooth				
It is as good as other games you played				
The game was challenging				

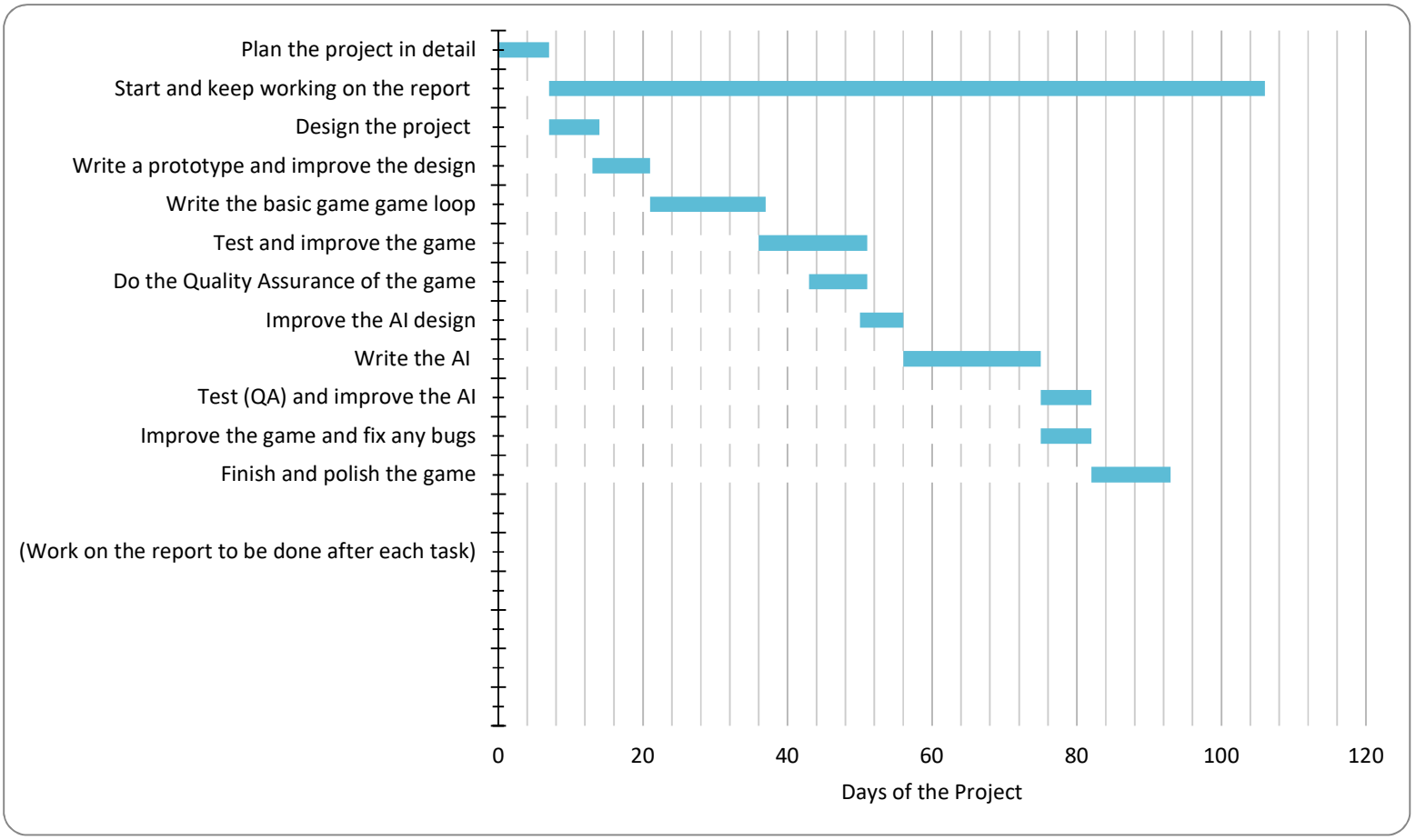
Table 2. An example of the second survey - Performance				
Question	**Comments**	**Agree**	**Neutral**	**Disagree**
Overall the game was stable				
I did not encounter any bugs or issues				
The enemy AI was smart				
Everything works as expected				

CHAPTER 5 - PLANNING

GANTT CHART

* = an automatically calculated cell

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)
First Sample Project				
Plan the project in detail	6/1	6/7	0	7
Start and keep working on the report	6/8	9/14	7	99
Design the project	6/8	6/14	7	7
Write a prototype and improve the design	6/14	6/21	13	8
Write the basic game game loop	6/22	7/7	21	16
Test and improve the game	7/7	7/21	36	15
Do the Quality Assurance of the game	7/14	7/21	43	8
Improve the AI design	7/21	7/26	50	6
Write the AI	7/27	8/14	56	19
Test (QA) and improve the AI	8/15	8/21	75	7
Improve the game and fix any bugs	8/15	8/21	75	7
Finish and polish the game	8/22	9/1	82	11
(Work on the report to be done after each task)				


Plan the project in detail
Start and keep working on the report
Design the project
Write a prototype and improve the design
Write the basic game game loop
Test and improve the game
Do the Quality Assurance of the game
Improve the AI design
Write the AI
Test (QA) and improve the AI
Improve the game and fix any bugs
Finish and polish the game
(Work on the report to be done after each task)
0
20
40
60
80
100
120
Days of the Project

REFERENCES

Bethke, E. (2003). *Game development and production*. Plano, Tex.: Wordware Pub., pp.3 - 36.

Botea, A., Muller, M. and Schaeffer, J. (2019). Near Optimal Hierarchical Path-Finding. *Department of Computing Science, University of Alberta*.

Bourse, Y. (2012). Artificial Intelligence in The Sims series.

Dillet, R. (2019). *Unity CEO says half of all games are built on Unity*. [online] TechCrunch. Available at: https://techcrunch.com/2018/09/05/unity-ceo-says-half-of-all-games-are-built-on-unity/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=Pg_iiAtjpWlznUbkoBpuPA [Accessed 4 Apr. 2019].

Edeki, C. (2015). AGILE SOFTWARE DEVELOPMENT METHODOLOGY. *European Journal of Mathematics and Computer Science*, 2(1).

Galyonkin, S. (2019). *Preliminary results for Steam sales in 2015*. [online] Steam Spy. Available at: https://galyonk.in/preliminary-results-for-steam-sales-in-2015-43bb49a767bd [Accessed 4 Apr. 2019].

Horti, S. (2019). *Why F.E.A.R.'s AI is still the best in first-person shooters*. [online] Rock Paper Shotgun. Available at: https://www.rockpapershotgun.com/2017/04/03/why-fears-ai-is-still-the-best-in-first-person-shooters/ [Accessed 4 Apr. 2019].

Lopes, G. and Kuhnen, R. (2007). *Game Design Cognition: The Bottom-Up And Top-Down Approaches*. [online] Gamasutra.com. Available at: http://www.gamasutra.com/view/feature/130542/game_design_cognition_the_.php [Accessed 4 Apr. 2019].

Nagy, Z. (2018). *Artificial Intelligence and Machine Learning Fundamentals*. Birmingham: Packt Publishing Ltd.

Orkin, J. (2006). Three States and a Plan: The A.I. of F.E.A.R. *Game Developers Conference*.

Rouse, M. (2019). *What is AI (artificial intelligence)? - Definition from WhatIs.com*. [online] SearchEnterpriseAI. Available at: https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence [Accessed 4 Apr. 2019].

Tricentis. (2019). *AI Approaches Compared: Rule-Based Testing vs. Learning - Tricentis*. [online] Available at: https://www.tricentis.com/artificial-intelligence-software-testing/ai-approaches-rule-based-testing-vs-learning/# [Accessed 4 Apr. 2019].

Ward, J. (2019). *What is a Game Engine?- GameCareerGuide.com*. [online] Gamecareerguide.com. Available at: http://www.gamecareerguide.com/features/529/what_is_a_game_.php [Accessed 4 Apr. 2019].

Wijman, T. (2019). *Global Games Market Revenues 2018 | Per Region & Segment | Newzoo*. [online] Newzoo. Available at: https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/ [Accessed 4 Apr. 2019].

Bungie.net. (2014). *Bungie Weekly Update - 06/27/2014 > News | Bungie.net*. [online] Available at: https://www.bungie.net/en/News/Article/11607/7_Bungie-Weekly-Update---06272014 [Accessed 6 Apr. 2019].

Poole, W. (2016). *Lionhead: The inside story • Eurogamer.net*. [online] Eurogamer.net. Available at: https://www.eurogamer.net/articles/2016-05-12-lionhead-the-inside-story [Accessed 6 Apr. 2019].

Sinclair, B. (2013). *GTA V dev costs over $137 million, says analyst*. [online] GamesIndustry.biz. Available at: https://www.gamesindustry.biz/articles/2013-02-01-gta-v-dev-costs-over-USD137-million-says-analyst [Accessed 6 Apr. 2019].

Suzuki, Y. (2015). *Yu Suzuki wants $10 million to make a truly open world Shenmue 3 • Eurogamer.net*. [online] Eurogamer.net. Available at: https://www.eurogamer.net/articles/2015-06-22-yu-suzuki-needs-usd10-million-to-make-a-truly-open-world-shenmue-3 [Accessed 6 Apr. 2019].

Dave, H. (2016) Rocket League: how the game's overnight success was a decade in the making [online]

TheGuardian.com Available at: https://www.theguardian.com/technology/2016/sep/06/rocket-league-psyonix-online-sports-game-football-motor-racing-combat

Association, T. (2019). *Laws of the game and FA rules*. [online] www.thefa.com. Available at: http://www.thefa.com/football-rules-governance/lawsandrules [Accessed 7 Apr. 2019].

BBC News. (2018). *Inside Rockstar: Red Dead Redemption 2*. [online] Available at: https://www.bbc.co.uk/news/av/technology-45896453/inside-rockstar-the-developer-behind-red-dead-redemption-2 [Accessed 7 Apr. 2019].

Campbell, C. (2018). *The Last of Us is worth returning to, 5 years later*. [online] Polygon. Available at: https://www.polygon.com/2018/6/8/17439466/the-last-of-us-ps4-play-e3-2018-sony [Accessed 7 Apr. 2019].

CD PROJEKT. (2019). *Strategy - CD PROJEKT*. [online] Available at: https://www.cdprojekt.com/en/capital-group/strategy/ [Accessed 7 Apr. 2019].

Docs.microsoft.com. (2018). *C# Guide*. [online] Available at: https://docs.microsoft.com/en-us/dotnet/csharp/ [Accessed 7 Apr. 2019].

Nintendo.com. (2006). *New Super Mario Bros. U Deluxe*. [online] Available at: https://www.nintendo.com/games/detail/new-super-mario-bros-u-deluxe-switch [Accessed 7 Apr. 2019].

Pitts, R. (2014). *How the team behind The Witcher conquered Poland*. [online] Polygon. Available at: https://www.polygon.com/features/2014/7/16/5884227/cd-projekt-the-witcher-3 [Accessed 7 Apr. 2019].

Technologies, U. (2019). *Unity - Manual: Unity User Manual (2018.3)*. [online] Docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/index.html [Accessed 7 Apr. 2019].

Technologies, U. (2019). *Unity - Scripting API: Sprite*. [online] Docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Sprite.html [Accessed 7 Apr. 2019].

Unity. (2017). *Behaviour Tree - Unity*. [online] Available at: https://unity3d.com/learn/tutorials/projects/2d-game-kit/behaviour-tree [Accessed 7 Apr. 2019].

King, R., Churchill, E. and Tan, C. (2017). *Designing with data*. Sebastopol, CA: O''Reilly Media, Inc.