UNIVERSITY
of
GREENWICH

An investigation into the Finite State Machine behaviour of an advanced game AI in the form of a Non-Player Character in a procedurally generated environment

FINAL YEAR PROJECT

Supervisor: Dr Tuan Vuong

WORD COUNT: 14392

A dissertation submitted in partial fulfilment of the University of Greenwich undergraduate degree programme

| BSc (Hons): Computing |

Abstract

The use of artificial intelligence within videos games has existed since the 1950's but has only truly become prevalent since the early 2000's. This project will investigate into the usefulness of an advanced artificial intelligence agent in a game environment which possesses semi self-sufficient behavioural tendencies based on the input it receives from the player and the environment. With the implementation of finite state machines in the diverse Unity engine this will be made possible and although, by definition it is not true artificial intelligence, it's one step closer to achieving full autonomy.

Acknowledgements

Firstly, I would like to thank my family who have been there to support me throughout the ups and downs that this year has presented, always offering advice and consideration with their words.

Secondly, I would like to thank my best friend who has a degree in animation, for assisting me in brain storming ideas for the visual side of this project because I am not artistically creative.

And last but certainly not least, I would like to sincerely thank Dr Tuan Vuong who has been a great supervisor and mentor to me throughout the duration of this year. Always pushing me to do better and when facing dilemmas, being pointed in the correct direction.

I could not have done it without all of the aforementioned people.

1 Table of Contents

2 Introduction

The purpose of this document is to provide a clear and concise argument which reinforces the claim made in the initial statement. This will be achieved by divulging into three sections which will increase the structural integrity of the arguments being presented. The first of which is a section on literature reviews where pre-existing published documents will be cited in order to make a strong argument and then critically analysed to either prove or disprove what is being said based on which will reinforce the thesis. Next a section on similar products which will look at already existing products which reside in the same domain as the one being created for this project. Comparisons and contrasts will be made between the available products to identify differences and/or superiorities in either side. And finally, a section which reviews the technologies being put to use in this project and why they were chosen versus the other available options.

The justification will be made clear by use of listing the pros and cons and providing a table which compares the best features from each technology which will be used in the end to design the prototype. The next section will include a detailed requirement analysis identifying the functionalities needed for the product to be successful, followed by the design documentation which will show how the product has evolved from stage to stage. Lastly, the document will be ended with a thorough analysis and evaluation identifying what went well and what could've gone better through the use of a table and a solid conclusion to assess the learning outcomes of this project.

2.1 Background

In referral to artificial intelligence regarding games, it is generally used to describe a broad set of algorithms but that does not define it because it also makes use of techniques from computer graphics, robotics and computer science in general. And when taking these aspects into consideration, video game artificial intelligence may not necessarily be defined as true artificial intelligence because there is not a set of criteria which states computer learning and other standard techniques must be used. Some level of automated computation is what truly defines artificial intelligence, not the level of complexity associated with a particular algorithm. And as a result, artificial intelligence has become more widespread and is present in almost every aspect of the video game industry and this project intends to put its robustness to use in order to demonstrate its usefulness.

2.2 Scope of this project

The intention of this project is to successfully develop and fully implement an artificial intelligence in the form of an NPC (Non-Player Character) which possesses the quality of self-sufficient behaviour through the implementation of Finite State Machines in a 2-Dimensional procedurally generated environment which means only one 'level' of the game will be manually created and using the design of this scene alongside C# scripts that are written, will procedurally generate an entire map of interconnecting scenes for the player to navigate and battle multiple enemies. The game will be showcased as a 2D game created in the Unity engine. This differs from a basic artificial intelligence which expresses no self-sufficiency in decision making characteristics because it has multiple behavioural states to choose from which will impact the way it reacts. These characteristics can be broken down into three subcategories which truly define a complex AI from something much simpler that doesn't possess behavioural tendencies.

Sense, the ability for the agent to detect (or is told about) things in their environment that may influence their behaviour. Think, the agent then decides on how it should respond to the thing it detected. Act, the agent performs an action which puts the previously made decision into motion. And now the three-step cycle is complete, and the situation has changed, therefore the cycle can be repeated with new data (Sizer, 2018). The game being created for this project shall consist of an advanced AI which has the ability to patrol within its construct without being given a path that it must use. It shall also possess the ability to decide when it should transition from the patrolling state to the aggro state which will allow it to chase the player and try to inflict enough damage upon the player, thus ending the game for the player.

This top down combat style type of game is still very popular to this day as similar games have been released in the past few years and have found extreme success bringing in millions of pounds. Concluding that with the right interpretation and approach, 2D games can be still extremely relevant in this day and age although majority of game developers have moved toward the 3D approach. This has had no bearing on the success of a well implemented 2D game which executes a great concept. The inspiration for this project came from the indie developer known as Edmund McMillen and his game 'The Binding of Isaac', which possesses similar characteristics to that of this project. Edmund became an overnight success with the release of 'Super Meat Boy', his first title, deemed "one of the greatest games ever created", followed by 'The Binding of Isaac' which had similar success (Glagowski, 2018).

The game will be created in the Unity Engine as a 2D project with the backend being C# and once implemented the project will be converted to a WebGL build which allows a project to be published on the web as a JavaScript program with the use of HTML5 technologies and the WebGL rendering API to run Unity content in a web browser, thus allowing the project to be accessible from a website that could accompany the game, making for simplified demonstration across any desktop platform which possesses a browser. To put it simply, the final product will be a Roguelike 2D Top-Down Dungeon Crawler style combat game.

2.3 Aims and objectives

- Research Report / Total = 8.0 days

A detailed and specific report containing in-depth information and research regarding the various elements of the chosen topic which shall allow for the completion of this project.

1. Initial proposal [2.0]
    - First document submitted presenting the initial project idea along with justification
2. Literature Review [3.0]
    - Citing other people's literature and reviewing it to build a supporting case
3. Final Research Aspect [3.0]
    - All research components brought together and combined for the final draft

- Design Documentation / Total = 4.0 days

This section of the project specifies the technologies used and the system architectures which shall be put in place during the implementation phase.

4. Requirement Analysis [1.0]
    - Identifying the high-level requirements which will be needed for the system

5. Statement of Requirements (Requirement Specification) [1.0]

    - Listing the actual high-level requirements identified during the previous stage

6. Paper Prototype [2.0]

- Designing a prototype of the game but on paper to help visualise the idea

- Implementation / Total = 35.0 days

At this point the Paper Prototype has been fully realised and implemented, fulfilling the requirement specification which was created during the previous phase of the project.

7. Base functionalities implemented/Realised Prototype [Total = 7.0 days]

- Paper prototype being adapted and reimagined as a real system [2]
- Base environment created within Unity [2]
- Base environment updated to become much more detailed and ready for the AI [3]

8. Full Implementation of Requirements [Total = 21.0 days]

- Basic Requirement Specs implemented [4]
- Basic Artificial Intelligence implemented [5]
- Artificial Intelligence improved to have more complex behaviour [8]
- Optimising the code to use less memory [2]
- Additional features added if there is ample time [2]

9. Blackbox/Whitebox Testing [Total = 7.0 days]

- Analysing and evaluating any immediate bugs found [2]
- White Box: Testing the internal structure and implementation myself [4]
- Black Box: Allowing an outside tester to examine the internal structure and implementation [1]

- Evaluation Report / Total = 2.0 days

A detailed report evaluating the overall outcome of the final product which should contain a reflection of what could've been done to improve the result.

10. Evaluation of Report [0.5]

- Evaluating the quality of the completed report

11. Evaluation of Product [1.0]

- Evaluating the quality of the overall product and the achieved outcomes

12. Evaluation of Personal Performance [0.5]

-Evaluating the work ethic and goals achieved over the entire duration of the project

2.4 Approach

When taking into consideration the vast number of methodologies and frameworks available for use, when referring to the subject of game development the most understandable approach would be the Dynamic Systems Development Method (DSDM). The reason for this is because of the flexibility that DSDM offers. Something like the Waterfall model could never be applied to a situation like this because there needs to be constant iterations of development and testing which repeat. When implementing this methodology game development begins with the creation of basic features which places the game at a useable and marketable state, and if not, at the very least reusable so that content never goes to waste regardless of whether or not the full requirement specification was met. The process of starting off with a basic design and progressively moving toward complexity is the organic flow that DSDM promotes.

The nature of the methodology allows for this as things are done step by step in an iterative motion thus allowing for the most important thing to always be implemented next in a high-level specification prioritisation. Building incrementally from firm foundations is the core of this approach. Finding the fun is a massive part of game development because there needs to be interesting features which attract the users, and with this methodology the users are actively involved during the development of the system thus making them more likely to embrace it and at the same time the constant feedback ensures that the system meets the requirements. With the implementation of this approach the system is more than likely to deliver the agreed specifications because of the type of workflow that coincides with this developmental methodology.

3 Literature Review

3.1 Approach to literature searching

The purpose of the literature review is to gather relative information from research reports, articles and journals published by others relating to the topic of interest in this project and review them thoroughly by analysing the in-depth details of how others have attempted to tackle the problem faced in this scenario. By doing so a greater understanding is created on how to avoid the potential errors which others have faced, and to either improve upon them or simply learn from them and take them into consideration during the implementation stage of development.

3.2 Identifying the problem

Initial research began in finding where artificial intelligence belonged in the domain of video games and how it's evolved into what it is today over many years of progression in its development. When deciding how to implement the game AI in this project it was first important to understand what the goal was because this determines the type of AI needed and how complex it should be. Seeing as this project is merely a proof of concept the AI could be created semi-hard coded with characteristics which were categorised by State Machine Behaviour, which means that the potential behavioural outputs were all predetermined in a behavioural tree where the AI has the ability to shift between these different states based on a series of clauses.

3.3 Artificial Intelligence in the domain of video games

Artificial intelligence (AI) has been relevant in the field of computer games prior to the actual terminology being coined. The first ever program that was designed to play chess was created in 1950 and was designed to test the computational complexity of the brand-new algorithm which was used against expert players but resulted in very little success. It was a few years later in 1956 where American computer scientist John McCarthy organised the Dartmouth Conference, at which the term 'Artificial Intelligence' was conceived.

Ever since the initial breakthroughs made during this era it greatly advanced the exponential progress that AI has made in terms of computational intelligence, algorithms and machine learning and this has resulted in agents with advanced intelligent behaviour in the form of non-player characters to be created which was another huge step in the right direction. (Westera, 2018)

In terms of what is seen in video games, it is usually a combination of Finite State Machines (FSMs) and Path Finding. The reason for this being is that in a typical game environment the player is expecting some kind of experience and the developer wants to know exactly what the player is going to experience but the algorithms associated with true AI being reinforcement learning, supervised learning, and unsupervised learning are unpredictable and will not guarantee a specific set of results which may be sought after. With a neural network that is constantly adapting and learning from the feedback that it's receiving from the player this leaves room for massive unpredictability because each unique players actions will result in the AI learning something different and whilst in some games this may be an appropriate approach where unpredictability is anticipated or expected. In most games this is not the preferred approach because there's too great of an opportunity for something completely unexpected to happen and from a technical standpoint it could be game breaking if the chosen algorithmic paradigm produces an output which isn't valid in the construct of the game environment.

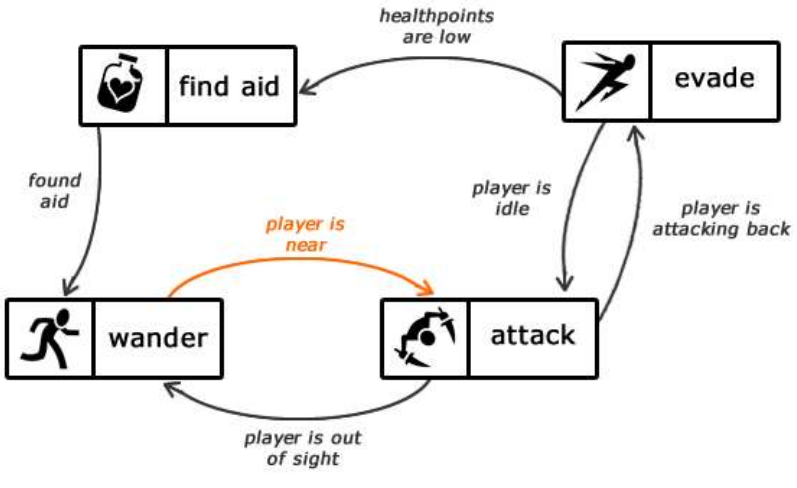
These predictable approaches have been classified as either automation or 'emergent gaming' by some researchers within the field but (Statt, 2019) and (Togelius, 2019) talk about the robustness of the usage of FSMs in game AI because the agent itself is not trying to directly clone human consciousness and behaviour which is the objective of true AI. But a sense of mystery that makes the game feel real is the feeling that developers aim toward achieving.

3.4 Finite State Machines and State Machine Behaviour

Finite State Machines (FSM) is a construct where a non-player character (NPC) can be in different behavioural states and move between them without direct instruction being hard coded within them. What this means is that the AI is assigned a sort of behavioural tree with certain behaviours being classified as states, and based on certain input the AI is receiving from the player and environment it is able to freely transition between these states such as idle to chase, chase to attack, etc. This is an extremely common approach toward video game AI in which some sort of intelligent behaviour is being created semi-autonomously. This is not to be confused with true AI which learns its behaviours from scratch or with very little assistance given at all. These make up some of the founding principles of video game AI.

"A finite-state machine, or FSM for short, is a model of computation based on a hypothetical machine made of one or more states. Only a single state can be active at the same time, so the machine must transition from one state to another in order to perform different actions.

FSMs are commonly used to organize and represent an execution flow, which is useful to implement AI in games. The "brain" of an enemy, for instance, can be implemented using an FSM: every state represents an action, such as attack or evade". (Bevilacqua, 2013)



These different behavioural states are influenced by two main factors. The first of which being positioning in the sense that a scene within an environment is made up of a certain number of vectors and in order for the agent to transition from the idle state to the chase state, the player

would need to be within a certain radius of the agent so that it is able to detect its coordinates via the vectors and from that initial analysis it will perform a calculation where it decides that it needs to transition from the idle state to the chase state because the goal is to catch the player and inflict enough damage so that the player loses the game. The second major influencing factor is the player. Without the player the agent will remain in an idle/patrolling state in which case no sensory actions can be performed by the agent because it needs a subject to interact with that will influence its decisions. If the player attacks the agent, it may choose to evade based on the amount of health it has remaining or it may continue to attack the player. All of these decisions are influenced by the input that the agent is receiving from both the player and the environment.

"Finite State Machines consist of four main elements which allow it to operate the way it does:

1. States which define behaviour and may produce actions.
2. State transitions which are movement from one state to another.
3. Rules or conditions which must be met to allow a state transition.
4. Input events which are either externally or internally generated, which may possibly trigger rules and lead to state transitions.

A finite state machine must have an initial state which provides a starting point, and a current state which remembers the product of the last state transition. Received input events act as triggers, which cause an evaluation of some kind of the rules that govern the transitions from the current state to other states. The best way to visualize a FSM is to think of it as a flow chart or a directed graph of states" - (Brownlee, 2002)

This approach will be extremely practical for the objectives that are going to be achieved in this project because of how simple and efficient this method can be when put to practice. Creating a dataset for a reinforcement learning algorithm could take weeks, if not months to achieve properly and may not even yield the results expected such as input handling, player handling, UI and progression. All of which are achievable through the implementation of FSMs where exact behaviours are predetermined, and the only possible unknown variable is when the transition of states may occur with the agent but this is one of the components which adds to the mystery of playing a game versus an intelligent agent. Some level of unpredictability in a controlled environment is desired.

3.5 Procedurally Generated Environments

"A major evolution of game development over the years would have to be the use of procedural generation. Instead of having a linear experience that is one-and-done, you can create something that always keeps the player guessing. Rogue-likes, survival and simulator games have been using procedural generation to extend their replayability. When it works, you have a game with almost unlimited replayability." - (Bycer, 2016)

In laymen's terms, procedural generation is an evolution of random generation in video games. Which means that specific content is not occurring in a repetitive order each time and there's some level of randomness to the environment each time the player starts a new game in order to create the most interesting experience possible. It ranges from specific items that can be found in different areas of the game all the way to the order of the level of environments.

There are many benefits that this methodology can offer if the developer knows exactly what aspects of the game they want randomly generated which can save time in terms of the development stage, and also maximise the amount of enjoyment had from the experience of playing the completed game and encountering new items/obstacles each time. But there is a list of limitations that this methodology offers and fortunately for what is being achieved in this project none of the existing limitations will harshly be applied. But they are as follows:

1. Inability to process uniqueness

Lateral thinking can not be accomplished with this method because there are a given set of rules that have to be obeyed and nothing can be procedurally generated outside of that which means the realm of possibilities is confined and this leaves little room for surprising modifications to be made.

2. Limited Foundation

The way that the player interacts with the environment needs to be simplistic in order for the environment to be replicated whilst having all the existing game mechanics still work. The more complicated the set of game mechanics are, the more difficult it will be for the environment to be procedurally generated and work flawlessly.

3. Handcrafted VS Mass-Produced

The final major criticism is that when a particular scene or environment is procedurally generated, and mass created it lacks in the handcrafted touches if unique attributes are trying to be added. With linear procedural generation this problem doesn't occur because scenes are intended to look the same with the exception of potentially having different types of enemies or items spawning in each room. But when trying to create an algorithm which adds a unique feel to a procedurally generated room it's almost impossible as there is a certain handcrafted essence that a computer is unable to capture.

In terms of the outcomes being achieved in this project this method of procedurally generating levels is perfect to add a bit more depth to the duration of a game so that a player isn't encountering let's say a single enemy AI, but there is the opportunity to encounter multiple enemies across different levels to complete the game.

3.6 Conclusions

The core focus of this literature review was to identify the type of artificial intelligence needed to obtain a minimum viable product whilst having limited experience in the field. Although there were multiple attractive approaches which all had potential to be successful, the approach being taken is definitely one of the most used and most realistic in this style of game and according to what is being proved in this project it works. There were two primary approaches which needed consideration, one was to make the AI fully autonomous with a machine learning algorithm, and the other approach was to implement Finite State Machines which is semi-autonomous AI, and considering the time constraints and difficulty of the subject, not to mention limited experience, the obvious choice was FSM.

4 Product Research

4.1 Comparing of similar products or platforms against a set criterion for Usability purpose.

4.2 Introduction

The reason for creating this section is to identify similar software to that which is being developed for this project in order to recognise where any inspiration may be coming from and to also form a set of criterion for usability purpose which can be compared and contrasted against each other to show how the software being developed in this project is better or worse and in what aspect it can be improved. Creating a more in-depth insight into similar products can also aid in seeing how this project can be taken further regarding future developments

4.3 Product 1 – The Binding of Isaac

The Binding of Isaac is a rogue-like, top down, 2D game where a lot of inspiration was drawn from in the creation of this project. The aim of the game is for the player to navigate through a procedurally generated environment whilst fighting AI NPC's until eventually reaching a boss room where the player needs to defeat the final boss in order to progress to the next level and this is the basic nature of the game. Each room within this game presents the player with unique NPC's to face and this is where the uniqueness of this game comes into play. The Finite State Machine behavioural trees are unique with each different type of NPC within the game which means they all react and respond differently when coming into contact with the player depending on multiple different variables to do with both the player and the environment basically making each experience within the game unique because each time the game is restarted the procedurally generated scene is randomised based on the usage of C# scripts therefore changing the route needed to complete the level and changing the type of enemy NPC's that are going to be encountered.

4.4 Product 2 – Super Meat Boy

Super Meat Boy is also a 2D game except instead of being in a top down style it is a side scroller which means the player progresses either left or right infinitely as the screen follows the player until the level or game is complete. The levels in this game are created statically and do not change when the game is restarted because they are more challenging, and the purpose of this game is built around problem solving by traversing through complex environments instead of plainly killing enemy AI NPC's. In this game there are enemy AI too, but the purpose is to avoid them and navigate around them whilst figuring out the level. The Finite State Machine behavioural trees of the enemy AI NPC's in this game is fairly simple and they are hard coded specific instructions regardless of where the player is in relation to the NPC (at least in most scenarios), but there are a few exceptions in the more complex levels.

4.5 Usability Heuristics

The purpose for identifying the usability heuristics which should be abode by with the creation of video game software is to make sure it is being created in accordance with an extremely vital set of criteria which ensure the success of the developed software in terms of stability, user-interface design and many more important characteristics. In doing so the products presented above can be fully analysed to the furthest extent because inspiration has been taken from both toward this project.

1. Visibility of System Status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time. This allows the player or user to always be aware of what they should do and what they should avoid doing because they receive constant updates from the system. For example, regarding this project when the enemy AI NPC attacks the player it is important that the player is notified on how much health is remaining before the player is defeated and the game needs to be restarted

2. User Control and Freedom

It is important that the user is given sufficient control over the environment and their player appropriate to the style of game. For example, if the player is progressing through the level trying to find the final boss so that they can complete the game, if they realise that they have gone the wrong direction into an incorrect room, it is appropriate that they should be allowed to go back and return to the correct course of action instead of being forced to progressed to a dead end. Also, sufficient user control can also come down to something as simple as player movement speed versus enemy AI movement speed. If the enemy is too fast for the player to escape the game becomes almost impossible to complete and so something such as speed needs to be relative or at least fair for the player.

3. Consistency and Standards

No matter the creator, this style of game always comes with very similar if not the same expectations of what buttons should do based on prior user experience. The WASD keys are always associated with movement and in games which only utilise the keyboard without the mouse then the arrow keys would be used for attacking/shooting where possible (dependant on style of game). It is important as a developer to operate in accordance with this norm within video games in order to satisfy user expectations to ensure the best experience possible. Design standards exist across the entire industry.

4. Error Prevention

"Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action."

No system nor player is perfect and both are capable of errors, so it's important to have error prevention strategies in place. For example, before closing a game there should be a popup which reconfirms the action and asks the player if they are sure they would like to close the

game, before just immediately proceeding to the action in case it was accidental on behalf of the player. Regarding the game AI, there are also mitigation strategies for the sake of error prevention. For testing purposes and debugging there is a shortcut key which controls the behavioural states of the AI and can manually force it to switch states in the case of an error such as the enemy went out of bounds.

5. Aesthetic and Minimalist Design

The overall appearance of the game and its interfaces contributes massively toward player attraction and whether or not the player shall remain interested. Simple practices such as keeping the game clutter free and making all information easily accessible and aesthetic will help maintain the level of interest that is generated from aesthetics alone. This also includes things such as the instructions on how to play the game and how it should be kept short and precise and not a overwhelming amount of unnecessary information.

(Nielsen, 2019)

5 Technologies used

5.1 Comparing and contrasting the available technologies

5.2 C# versus C++

The game development engine chosen for the creation of the prototype side of this dissertation is Unity. The reason for this being because of the vast advantages that Unity offers for beginner game developers. Although that alone is good enough reason to justify the decision, it goes much further than that and the reasonings will be justified.

The biggest difference between the two is the programming languages associated with each form of software. The Unity engine makes use of Microsoft's C# which is a component-orientated programming language and the Unreal engine makes use of C++ which is an object-orientated programming language. What this means in terms of programming is that when working in C++ the focus is drawn toward the associations between classes that link together to form a large binary executable. Meanwhile a component-orientated language doesn't require you to know its inner workings because the language itself is built around exchangeable code modules that work on their own.

The major differences between C# and C++ are:

1. "C++ compile's into machine code, while C# compiles to CLR (Common Language Runtime is the virtual machine component of the .NET Framework), which is interpreted by ASP.NET.

2. C++ requires you to handle memory manually, but C# runs in a virtual machine which can automatically handle memory management.

3. C# does not use pointers, while C++ can use pointers anywhere.

4. C++ can be used on any platform, though it was originally designed for Unix-based systems. C# is standardized but is rarely used outside of Windows environments.

5. C++ can create stand-alone and console applications. C# can create a console, Windows, ASP.NET, and mobile applications, but cannot create stand-alone apps."

(CSharp-Station Team, 2018)

5.3 Unity versus Unreal Engine

From the perspective of a beginner developer the obvious choice to make is to go with Unity because of the fact that the memory is managed for you and that is a massive part of game development. Memory management is closely related to optimisation because for every line of code that is written it needs to be optimised in such a way that once executed the application does not consume all of the resources that the computer has to offer. It needs to be effective yet efficient and Unity offers this because of the fact that the language associated with it is C#. So, now that it has been established that C# is the superior language from a beginner's perspective, we can take a look at the other attributes which make Unity a great choice. Besides the programming itself, some of the key features about Unity that make it stand out is the support available, tutorials, and the gigantic asset store.

Unity's community is filled with qualified developers who are readily available to assist beginners with any issues they may be facing with their projects and help get them on track which can be extremely frustrating when encountering bugs and errors. There's also a wide variety of certified tutorials available which explain how to do certain things within the engine such as scripting, creating new environments, etc. And lastly the asset store, it is filled with a huge number of items which can help speed up the development process by importing things that may be a waste of time for a developer to create when they should be focusing on the programming aspect.

The Unreal engine although similar to Unity in multiple ways doesn't offer as many benefits. It is more graphically intensive but that has no bearing on the outcome which will be achieved from this project because the focus isn't on graphics, but on the artificial intelligence agent being programmed.

Unreal offers a variety of blueprints which is essentially the building blocks and foundation of any project and to use these requires no experience with any programming and although some beginners may view this as a pro, it's actually a con because it's encouraging the user to not learn any programming even though it will become essential in the completion of any project.

The following table will compare and contrast the key major features and differences between the two chosen products of comparison.

	Unity	**Unreal**
Automatic memory management	Yes	No
Asset Store	Yes	Yes
Certified tutorials available	Yes	No
User-friendly UI	Yes	No
Fast rendering	No	Yes
Extremely graphicly intensive	No	Yes

6 Legal, Social, Ethical and Professional Issues and Considerations

In terms of legal issues associated with this project, the key factor that needs to be taken into consideration is Intellectual Property Rights (IPR). Stealing another individual's idea which has already been created can breach copyright law as it's theft of intellectual property. In this scenario there is inspiration from other games but no direct plagiarism and/or theft of intellectual property. If the game were to consist of gambling aspects which involved real currency, then it may also be in breach of the law depending on the audience it's set out for. There is a fine line between social gaming and gambling which may not be crossed, but for this project there is no use of in-game or real-world currency. The system of formal self-regulation in this industry is known as PEGI (Pan-European Game Information), and it ensures that the targeted audiences are kept safe offline and online by setting appropriate age restrictions.

PEGI has a guideline which needs to be followed and kept in accordance with, in regard to advertising as well, based on age restrictions certain games may only be advertised in a certain way and on specific platforms. In the context of this project all relative laws will be abided by. Video games can have a meaningful social impact but in this case the design and implementation are relatively simple compared to existing Triple-A game titles which may dramatically influence the way a person thinks, thus preventing the possibility of addiction which can lead to social ineptness. Interpersonal abilities may be impacted as an individual can spend so much time in a game that they lose touch from reality and begin living a secluded lifestyle. In regard to this project these described circumstances will most likely not be an issue as the end product is a game which can be completed in a mere matter of minutes.

Ethical issues in regard to the video game industry is a massive topic which can be summarised in a few concise points and although it's relative to all games no matter the genre, it's very much so based on the type of content existing within the game. The key issues are with violence, sex, rating, stereotyping, addiction within the community, and bad social habits such as smoking, drinking or even the consumption of drugs. The major concerns with the aforementioned topics are that they desensitise players and can change their perception of what they believe is acceptable and this leads to a generation of individuals potentially making questionable life decisions based on the influence of a game. In regard to this project, because of the simplicity

of the overall concept and design these factors mentioned above will not play a role. The only issue that may apply is a minor display of violence because in the 2D game there will be Non-Player Characters (NPCs) which chase after and attack the player to prevent him/her from achieving the end goal which is navigating to the final level and defeating the boss which will commence the end of the game.

No graphic displays of violence will occur, and the overall artwork style is retro and pixelated which are child friendly. The professional issues are very much so integrated with the ethical issues in regard to this industry which have been covered in detail above. One subject worth mentioning is "The Independent Games Developers Association (IGDA) which is an organisation that supports people making their games independently, they have been assisting all types of game developers from coding to script writers. Their mission is to advance the careers and enhance the lives of game developers by connecting members with their peers, promoting professional development and advocating on issues that affect the developer community." - (Edwards, 2012)

This article is of interest because it demonstrates the type of professional bodies that exist within the industry that support indie (solo or part of a small team) developers and they may be of great use in assisting indie developers in establishing a footprint in the world of game development. If this project is a success, then an organisation such as the one mentioned may be of great use.

7 Requirements

7.1 Analysis of requirements

After extensive and thorough research into what is trying to be achieved with the implementation of this project, it was made clear what needs to be achieved in order to declare this project a success by determining the needs and conditions which should be met.

This began by breaking down the crux of this project into MoSCoW prioritisation so as to attain a more finite understanding of what needs to be done first to achieve a minimum viable product (MVP), and this initiated with the minimum behavioural tendencies that the AI was capable of which was the differentiation between a purely hard-coded NPC and something which reflected some level of artificial intelligence as a proof of concept.

Once the characteristics of the AI had been determined and prioritised in order of what needs to be implemented first, then it was possible to move onto the player which is controlled by the user of the system and to see to what extent the player is able to interact with the AI and how other variables such as time, distance, speed and how the environment all play an important role.

7.2 Functional requirements

Entity:	**Requirement:**	**MoSCoW Prioritisation:**
AI	Is able to chase the player	M
AI	Is able to wander between different waypoints	M
User	Is able to control player	M
User	Is able to escape the AI as player	M
AI	Is able to idly stay in one position	S
User	Is able to attack the AI as player	S
AI	Is able to attack the player	S

Environment	Is able to procedurally regenerate in a singular formation	S
User	Is able to kill the AI as player	C
AI	Is able to kill the player	C
AI	Is able to enter a different scene	C
User	Is able to enter a different scene as player	C
User	Is able to save the game as player to continue progress	W
Environment	Is able to procedurally regenerate in different formations	W

7.3 Non-functional requirements

Entity:	**Requirement:**	**MoSCoW Prioritisation:**
Maintainability	The code written for the game must be maintainable, updatable and efficient. Therefore, it has been created in C# where memory automatically manages itself due to the nature of the language thus keeping it optimal	M
Usability	Each key press in the game will relate to an action performed by either the player or the AI	M
Response Time	The average response time between key press and reaction must be less than 0.5 seconds to ensure responsiveness	S
Required Resources	The game must be able to operate on a system which has a minimum of 2GB of RAM and it must take up less than 1GB of storage space	S
Frame Rate	The game is set to achieve no less than 30 frames per second as long as the computer in question can graphically support it	S
Platform	The game must be able to run on any Windows based system with a requirement of Windows 7 or above	S

8 Design of the Application

8.1 Introduction

This section is one of the most important parts of the project because it outlines the way certain things should be done and starts bringing concepts together to form a concrete image that can be used during the development stage. A high-level understanding needs to be created in order to begin development on the product because this is the only way to ensure success, and every statistic points toward the fact that projects which are planned in advance with thorough depth usually end in success or at least in achieving a Minimum Viable Product.

This section will go into detail explaining both major aspects of the product, being the frontend which is what the user sees, and the backend which is core content which holds everything together. Along with an overview of the entire architecture behind the system which explains how all the entities within it react to different situations.

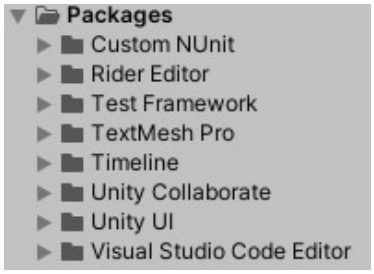


Figure 1 - Packages required for the project to function within Unity

From the way the project was designed within Unity, to setup and initialise it on an external machine will require the following packages to be installed in order for the project to perform all of its functionalities correctly.

8.2 Initial Planning

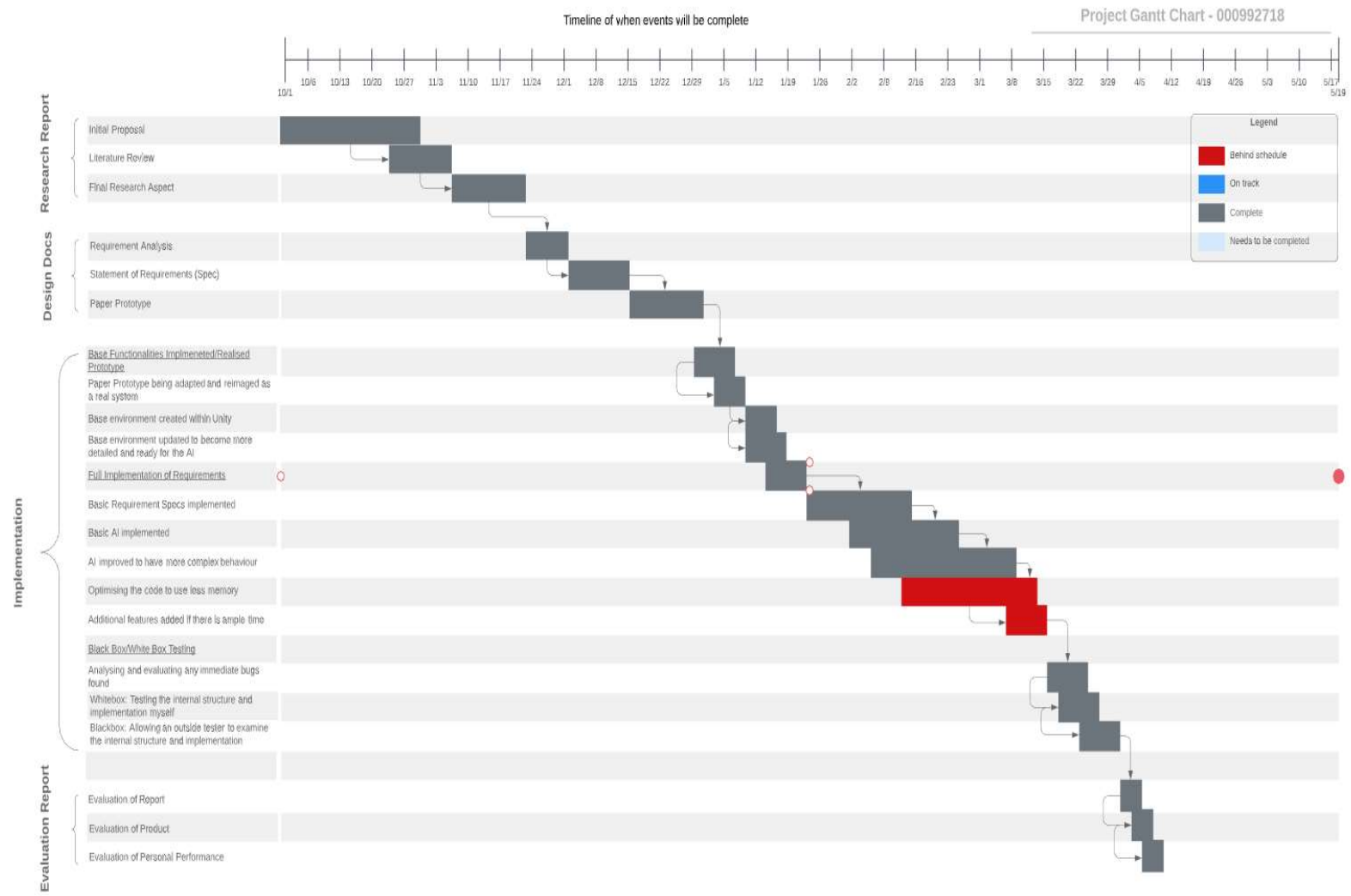


Figure 2 - Gantt Chart displaying project projections

Planning initially began with the creation of a Gantt chart which put everything into perspective of what needs to be done and when. This was an extremely crucial step because a project like this has many avenues which need to be explored and if the correct amount of time is not allocated to each aspect then the minimum requirements for the system will not be met, thus rendering the project a failure.

The purpose of this was just to get an idea of the bare skeleton of the system as a whole by setting deadlines for when certain aspects need to be achieved in order to stay afloat.

8.3 Overview of the Architecture within the System

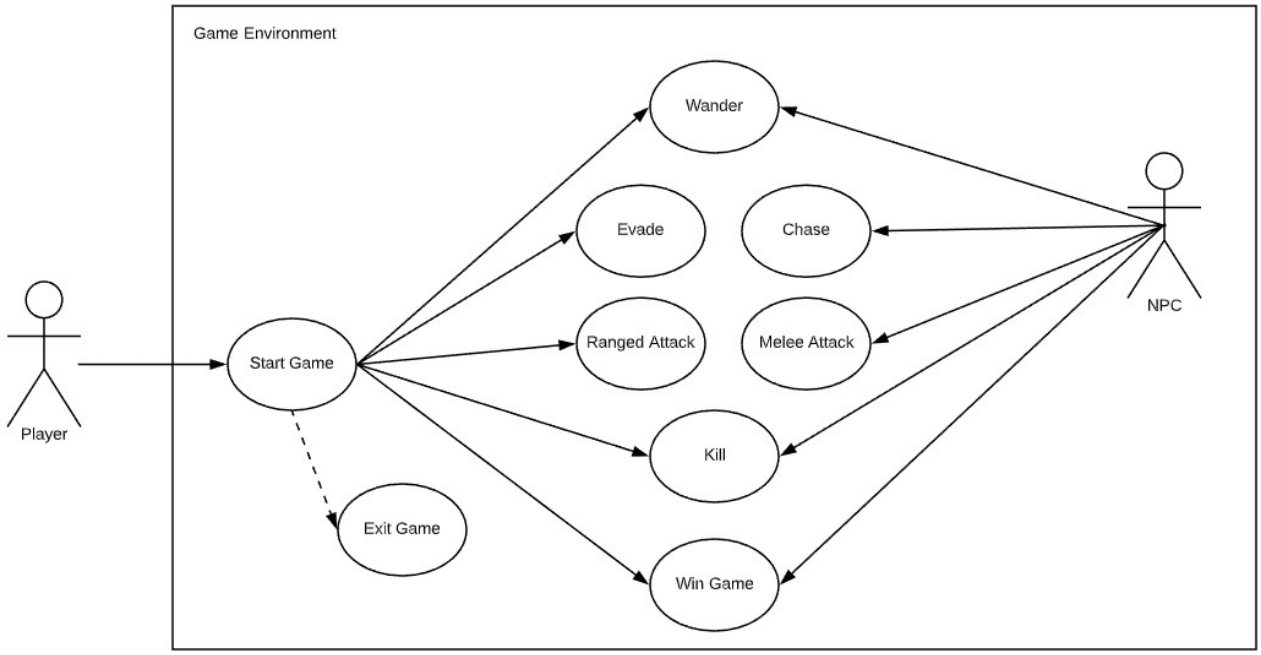


Figure 3 - Use Case model of system

A basic UML Use Case diagram has been used to create a basic understanding of the entities that exist within the system, and how they can influence one another by performing various actions that they are capable of in order to achieve an overall outcome. This has been displayed with a graphical representation because it easily describes the problems being faced in an understandable way to any person who has no prior knowledge of the system.

Once the user begins the game there are several actions they can perform displayed above in a hierarchy from top to bottom describing the order in which they should be done but there is no particular fixed order in which they must be performed, the user is free to decide. But the player should begin by wandering the environment to find the enemy NPC because the only way to finish the game is by defeating it.

Once the player has made contact with the NPC, they can then decide whether to evade or attack, if they chose to evade the NPC will continue to chase until it has made contact with the player and is able to attack it, or until the proximity needed to chase has been exceeded and the NPC will return to a state of patrolling. But the final outcome of the game will always result in either the player defeating the NPC or by the NPC defeating the player and then the game would need to be restarted for a second attempt.

8.4 Interface Design

Figure 4 - Screenshot of interface from the game environment

This screenshot was taken directly from development of the interface in debug mode so that is why the gridlines are present. Although the concept of this project was to prove the capabilities of the behavioural tendencies behind the NPC (blue avatar on the right), to some extent this is still a game at its core and therefore required some level of visual attractiveness because aesthetics are important too and for example a slightly visually impaired person could find it a lot easier to view something with bright and vibrant colours over an interface which is dull and colourless.

Nielsen's Heuristics for User Interface Design

Visibility of System Status – For example, when the player takes damage from the NPC, they are informed of how many more times they can incur damage before the game is lost, in the form of a health system. This is in support of the Nielsen's rule that states, "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time." (Nielsen, 1994)

User Control and Freedom – Within the game environment the player is freely able to do as they please without direct instruction and if they encounter a bug or have made a mistake they are also able to restart the game, which supports the Nielsen's rule which states "Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue." (Nielsen, 1994)

Aesthetic and Minimalist Design – The interface only includes crucial information for player to know that makes them aware of the possibilities within the environment, there is no clutter of useless information, therefore the game interface has been kept aesthetic and minimalist. This is in support of the Neilsen's rule that states, "Prioritization comes to play when this aspect is being considered. For the designer or the developer, all the information that's being presented on the page is relevant." (Nielsen, 1994)

8.5 Backend Design

The design for the backend came in 3 different stages. First the environment was designed, then the player, and lastly the core of the project which is the AI NPC.

The Environment:

When designing the environment it was important to take a few things into consideration like the dimensions, because a certain amount of space is needed to properly test the behavioural capacity of the NPC. Another important thing to do with the actual backend side of the environment is colliders. The scene eventually comes to an end at some point and collision parameters need to be added to both the objects within the environment as well as the player and the NPC to make sure they cannot walk out of bounds, for example. In terms of a published game this would be an extremely crucial thing to implement but considering this is merely a prototype the colliders were not fully debugged as time became an issue. As a result, there are occasionally errors regarding the colliders which prevent them from functioning correctly but in general they work.

The Player:

Designing the player was one of the simplest aspects of this project because there is so much readily accessible information to do with player configuration within video games, and one of the most important aspects to take into consideration was consistency. A player in this case would expect certain controls to perform certain actions depending on the type of game and/or environment. For example, the WASD keys are almost always related to movement and depending on the style of game, the arrow keys would be used for attacking. Therefore, it was important to program the player in this same way which conforms to the typical design considerations of this particular style of game.

The NPC:

The design considerations were tough when regarding the AI NPC because there were so many possible ways of approaching the subject and with limited knowledge and experience on the exact topic it became increasingly difficult. After thorough research it was made apparent that there were two potentially good ways of approaching the subject, either with a reinforcement learning algorithm, or with the usage of Finite State Machine algorithms which both represent their own sets of strengths and weaknesses depending on the end goal.

Reinforcement learning (RL) algorithms are especially useful when time is not a consideration, because multiple simulations need to be run simultaneously to achieve the optimal result and there is a lot that can go wrong. The reason for this being that RL operates with a reward system for the AI when it achieves correct outcomes, and this can lead to the AI creating tendencies of making the same choices even if it gradually leads to negative outcomes. And then the reward

system and algorithm need to be tweaked and the training process begins all over again. This was not an option regarding this project because time was not a luxury and so something that would definitely work needed to be implemented and this is where the decision was made to implement a Finite State Machine algorithm. A detailed explanation for this decision is included in the development section of this report.

9 Development of the Application

9.1 Introduction

This segment of the report primarily focuses on the actual application that was developed and implemented and the steps that occurred to achieve this. It was done by creating several prototypes which were each responsible for a milestone being reached during this project and by achieving several milestones it brought the entire development process closer to a minimum viable product until something was accomplished which proved the concept. Although there were two key prototypes involved in the creation of this project, the core programming behind each was mostly similar and so it has been discussed in detail during this section where the prototypes have been broken down. In this section the programming will be discussed in depth and the primary differences between each prototype will be noted and explained below.

9.2 Prototype 1 - The Environment - Description and Programming

This prototype primarily focused on the creation of the game environment that the AI would exist inside of and getting it to a level where it was ready for real world testing with a player and the AI NPC. But a lot of steps occurred before the testing period occurred and it will be unveiled in this prototype's explanation. The first decision that needed to be made is what this environment would look like and if it would only consist of a singular 'level' so to speak as a test bed, or if there would be multiple layers to this environment in the form of numerous scenes making it easier during the testing phase. And this is where the idea for a procedurally generated environment came from. What this means is that only one scene was designed and with the usage of C# scripts it was possible to make this scene reoccur in almost any designed formation.

How this was achieved is by first creating the scene itself with objects known as 2D Sprites to make up the floor, walls, and doors. Once it was physically there, a series of C# scripts were attached to the room which would operate as a room controller for when the scene was ready for testing with the NPC and the player. Within the room script a method was added known as OnDrawGizmos() and using this the parameters of the scene could be manually defined so that when the script was ready to procedurally generate the rest, it would know the exact dimensions it needs to copy.

Secondly, within this same script a method known as GetRoomCentre() which makes use of public Vector3 contained a calculation to obtain the exact co-ordinates for the centre of the room so that when the script was ready to procedurally regenerate, it now had the information pertaining to the exact dimensions of the scene, and also where the centre point of each room was so when doing the calculations to replicate each room and put them side by side it would do this using the centre point of each as a point of placement before calculating the appropriate distance needed from the perimeter of each room to have them side by side.

Finally, in the room controller script using the co-ordinates of each room as a means of identification, this information was added to a list to be accessed by a later method. Next, in the void Start () method of the script which is initialised the moment the game is started, a predefined number of rooms was noted so that once the script is activated to procedurally generate multiple rooms, then it wouldn't exceed this predefined number. And lastly, the RoomQueueData() and RoomLoad() methods were created which contained similar IF statements, the first stated that it should keep adding rooms to the queue of objects waiting to be initialised until the maximum number is reached, which was the predetermined number. And the second was an IF statement to return null once this action has been completed so that it was not an infinite loop causing a potential memory leak. There are slightly more finite details pertaining to how the procedurally generated scene operates but all of the core functionality has been explained.

9.3 Prototype 2 - The Player - Description and Programming

Once the game environment was setup and debugged for the most part. The creation of a second prototype was now necessary where some of the core factors of this dissertation could be implemented. Considering that the main focus of this entire project is the way that the NPC reacts with the player and the environment, before even creating the NPC, the player needed to be made first because it would be impossible to test the key characteristics of the NPC such as its ability to chase, and attack without a player existing within the environment.

And so began development on the player which was a rather simple process because it is quite literally a 2D Sprite used as the player avatar, and the glue holding it together is the player controller script. Within this script a few crucial variables exist which determine how the player will react with the environment. This being a public float speed variable which calculates the speed at which the player is moving. A public Vector2 named MovementVelocity which determines the reaction time and responsiveness of the player letting go of a movement key and seeing how much further the player moves before grinding to a halt. This feature determines how snappy player movement is and for a game like this it's an important thing to take into consideration. And lastly, a RigidBody2D which is a method that controls the physics for 2D Sprites needs to be applied to the player avatar as this is the most important element in determining how the player will react with the environment.

The next step was to implement a method that could fetch movement input and apply it to the RigidBody2D which is bound to the player 2D Sprite. Considering that movement is of a nature that constantly needs to be updated, this function was placed inside the void Update() method which is continuously sending and receiving updates every moment that the game is initialised. In this function a new Vector2 was declared which makes use of attributes relating to the second dimension which is the format of this project, and it fetches input from both the horizontal and vertical axes. Then to make use of the MovementVelocity variable this movement input is multiplied by speed to get a rate at which the player can move.

And the final thing to be done in terms of player movement is to add the Player.position with MovementVelocity and multiply it by Time.fixedDeltaTime. Now the script can be attached to the player. The finishing touch was to add a 2DBoxCollider to the player so that it was capable of collision because it would be important at a later stage, and to setup the key bindings relating to player movement which were the WASD keys.

9.4 Prototype 3 - The AI NPC - Description and Programming

With both the environment and the player both fully setup, it was now time to begin development on the AI NPC which is the core of this entire project which brings everything together. There are 3 primary scripts associated with the NPC which determine the functionality behind it's capabilities but what determines the way in which the NPC decides to use these functionalities is the Finite State Machine behavioural tree which was created in the hopes of achieving a semi-autonomous artificial intelligence because this function controls the scripts. As mentioned previously, there are 3 core scripts associated with the NPC which will all be discussed in depth below. That being a PatrolBehaviour script, an IdleBehaviour script, and a ChaseBehaviour script.

The PatrolBehaviour script is what determines how the NPC is able to patrol almost aimlessly around the scene until it comes into contact with the player. The way this was achieved is by first declaring 3 variables within the script. A private int named RandomLocation, a public float called Speed, and a private variable named speed patrol which stores its information to do with the previous point patrolled in an entirely different script named LocationsPatrolled.

Within the LocationsPatrolled script multiple vectors within the scene had been defined as points that the AI is allowed to patrol between and when that script was called within the PatrolBehaviour script it was assigned to a list within an OnStateEnter() method which is called when a transition starts and the state machine starts to evaluate the current state. Within this same method the RandomLocation variable was defined and given a Random.Range value of 0 so that when the NPC begins to patrol between these predetermined waypoints in the form of vectors it will have no preference thus allowing it to roam aimlessly which encapsulates some level of artificial intelligence. And then by using an animator.position to translate the position of the NPC and assigning it a speed by means of the same method used to determine speed for the player, it was able to patrol the scene.

Next was the IdleBehaviour script, this was primarily used as a means of debugging so when trying to get something right it was important to be able to force the NPC to stand still for as long as needed. This was achieved very simply by defining a key press which manually overrides the Finite State Machine behavioural tree and forces the inverse of the isChasing behaviour (the behaviour pertaining to the ChaseBehaviour script) parameter within the FSM, thus forcing the NPC to transition from whichever behavioural state it's in to the idle state where it remains perfectly still. This was achieved by the use of a singular IF statement within the OnStateUpdate() method which is called on each Update frame between the OnStateEnter() and OnStateExit() call-backs.

Finally, the ChaseBehaviour script was created and this script determines the way in which the NPC chases the player. This was achieved by first declaring a couple variables. The first of which was a private Transform called PositionOfPlayer which was responsible for updating the position of the player constantly, and the second was a public float variable called speed. Next, the PositionOfPlayer variable was defined by applying a method to it which transforms its own position based on the position of the player by locating the game object with the tag "player" assigned to it.

In the OnStateUpdate() method the same function that was used to determine the speed of the player was used once again, except this time the values changed, making the NPC move at a slightly slower pace than the player so that the game is actually playable and winnable from the perspective of the player. Within the same method two IF statements were created, both of which manually overriding the FSM with direct instructions to either stop chasing and start patrolling, or to stop patrolling and start chasing the player. Once again, for the purpose of debugging.

9.5 Logic behind Finite State Machine

Although the actual behavioural capabilities that the NPC is proficient in are defined within scripts, the essence of artificial intelligence within this project comes from the nature of the Finite State Machine and the way in which it operates. Technically speaking, it is a mathematical abstraction used to design algorithms, but in layman's terms it reads a series of inputs which all exist within different states and based on the content of the scripts the NPC is then able to utilise the FSM to freely transition between states which encapsulates a level of semi-autonomy.

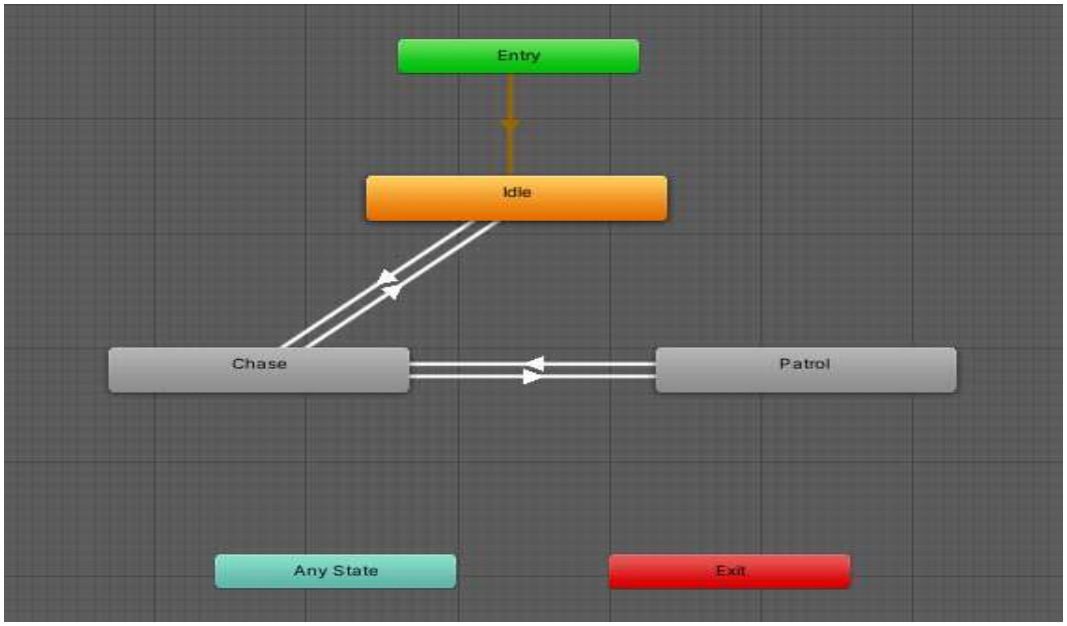


Figure 5 - FSM which represents the logic behind the NPC

Each script made previously relating to behaviour was applied to one of the states within this FSM and then the relationship between each state was given meaning by telling it how it can freely have a back and forth connection within itself to capture some level of artificial intelligence instead of forcing it to act as a tree which can only move in a singular direction.

Regarding both the player and NPC, in the early stages of this project a mistake was made involving the physics attributes applied to both entities in question. Their body types were set to kinematic instead of dynamic and this changes the way they are able to react to other game objects such as each other, projectiles, the environment, etc. Dynamic entities involve qualities such as mass, linear drag, angular drag, gravity scale, etc and none of these variables had been taken into consideration during the initial development stages and at the point of realisation it was too late to undo. And as a result, the player and NPC cannot physically harm each other as they are unable to collide, being on the same plane of existence.

10 Testing

10.1 Introduction

Considering the fact that the approach used for the implementation of this project is DSDM, this means that in order to stay in accordance with the methodology, the program would need to be frequently tested once various iterations of development have been completed because this is the nature of the methodology. In doing so, bugs that were encountered were more easily overcome because they could almost immediately be pointed out before it becomes submerged beneath a sea of more code. This is one of the largest benefits of the usage of this methodology and this section will describe the more finite details of how this method of testing came into fruition and the results alongside it.

A large part of this section was testing the integrity of the Finite State Machine which controls the transitions between the logic of the NPC. This was difficult because it is something which can only be thoroughly tested by means of physically playing the game dozens of times and waiting to see if the expected results occur due to the transitions in the FSM being tested.

10.2 Interim Testing

There is no documentation for this section because it was simply achieved by obeying the rules of the methodology chosen and putting it to practice. Considering the limited personal experience within game development there was room for many errors along the road of implementation and this increased the need for incremental development followed by thorough testing to debug the physical and logical errors encountered. Having a good relationship with my supervisor made this phase manageable because I could always be pointed in the right direction and receive good advice regarding debugging code and what to try and tackle next.

10.3 FSM Test Plan

Transition Testing:

There are a few different means of testing FSMs against each other but in a case where there is only one FSM which needs its integrity tested, the same measures of testing can be applied but the way it is validated is by performing the same tests multiple times within the same FSM so that the integrity of each individual state is thoroughly put through trial. These tests are as follows:

1. Output Error: Test 1 and test 2 differ in the output of a transition;

2. **Transference Error:** Test 1 and Test 2 differ in the final state. When the final state achieved in both FSM is different after applying a test case;

3. **Transition Error:** It is the general term for output or transference error;

4. **Missing States:** Test 1 has fewer states than Test 2;

5. Extra States: Test 1 has more states than Test 2.

(Pinheiro, 2014)

Whilst taking these 5 key points into consideration, testing began on the FSM test bed and after multiple iterations of debugging the success rate turned out to be almost 100%. The code is not 100% bug free and therefore is destined to encounter an error at some point but after performing the exact same actions within the game environment on 20 different occasions and receiving the same result from the FSM in 19 out of 20 cases, it can be assumed that the testing process was a success as the system had a success rate of 95% to perform the same output after receiving its regular input.

Below a table has been made to show the possible conditions that can be met and the effect it will have in terms of an output in both a behaviour and a transition of states to reflect the recognition of an action being performed.

Condition:		**Effect:**	
Current State	**Input**	**Output**	**Next State**
Idle (state 1)	Entry point for the NPC when the game is being initialised	-	State 2
Wander (state 2)	The NPC begins to wander randomly between waypoints	The state of wandering is only interrupted when the player enters a certain radius within the	State 3

		NPC	
Chase (state 3)	The NPC begins to chase the player	The player is chased until it is attacked and killed or until it can regain a certain distance from the NPC	State 4
Attack (state 4)	The NPC begins to attack the player	The player takes damage and either dies or evades NPC attack (hence the possible states it can transition to)	State 3 or State 4

As mentioned previously in the report, the nature of the Finite State Machine means that depending on the input that the entity in question is receiving, it is then able to transition to an appropriate state based on a set of guard conditions which is a Boolean expression. Guard conditions are a set of rules which govern the way a FSM is triggered. A diagram made custom for the system within this project is displayed below but in order to understand the rules which govern the way it freely transitions between states, a set of principles need to be understood:

1. A transition may have a guard condition, which is a Boolean expression.

2. It may reference attributes of the objects that owns the finite state machine, as well as parameters of the trigger event.

3. The guard condition is evaluated when the trigger event occurs.

4. If the expression evaluates as true, then the transition fires, that is, its effects occur; otherwise, the transition does not fire.

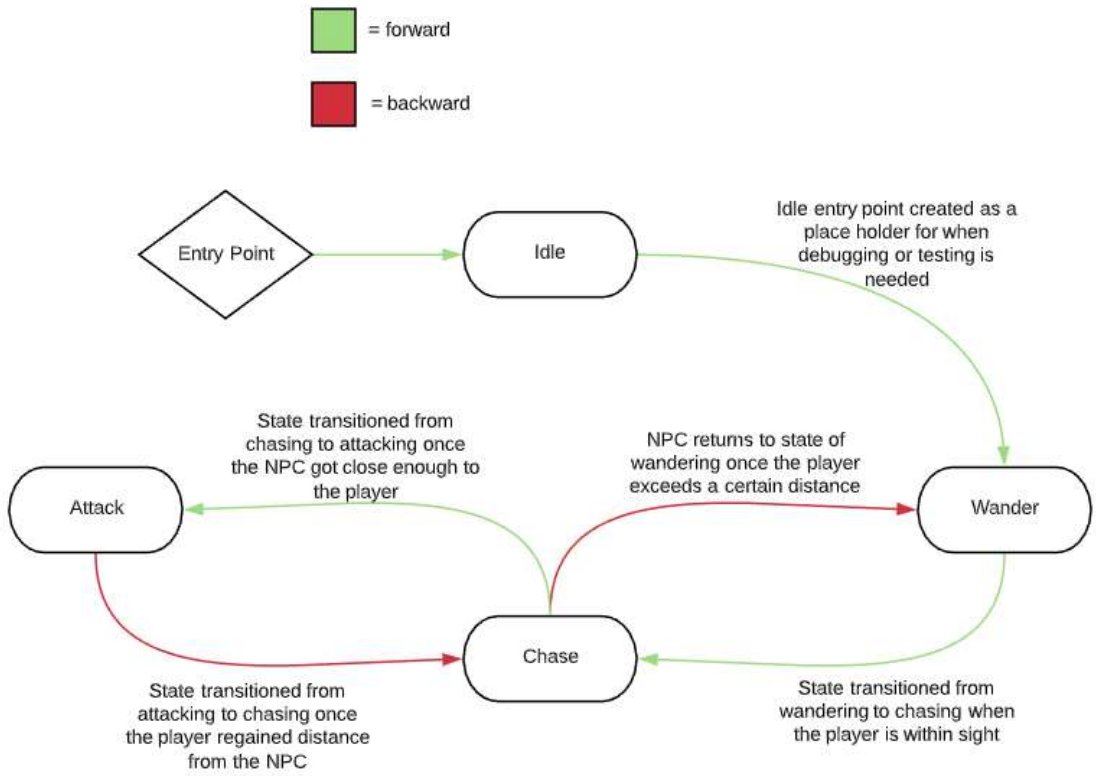


Figure 6 - Explanation of the logic which defines the NPC FSM

10.4 White-Box Testing:

Although the FSM test plan broke down the basics of how the integrity of the system is going to be tested. With everything complete and development coming to a halt, the individual functionalities performed by each entity need to be looked at under a microscope even if there isn't that many functionalities in the first place, because in order to establish what level of success the system as a whole has achieved, all of the features performed need to be carefully analysed.

To somehow make sense of the data needed to analyse the system, it has been categorised and tabulated for the sake of convenience and readability and the results from this testing can be found in Appendix B.

11 Reflection

11.1 Requirement Fulfilment

Functional Requirements:			
Entity:	**Requirement:**	**MoSCoW Prioritisation:**	**Achieved? (Y/N)**
AI	Is able to chase the player	M	Y
AI	Is able to wander between different waypoints	M	Y
User	Is able to control player	M	Y
User	Is able to escape the AI as player	M	Y
AI	Is able to idly stay in one position	S	Y
User	Is able to attack the AI as player	S	Y
AI	Is able to attack the player	S	Y
Environment	Is able to procedurally regenerate in a singular formation	S	Y
User	Is able to kill the AI as player	C	N
AI	Is able to kill the player	C	N
AI	Is able to enter a different scene	C	Y
User	Is able to enter a different scene as player	C	Y
User	Is able to save the game as player to continue progress	W	N
Environment	Is able to procedurally regenerate in different formations	W	N

Non-Functional Requirements:			
Entity:	**Requirement:**	**MoSCoW Prioritisation:**	**Achieved? (Y/N)**
Maintainability	The code written for the game must be maintainable, updatable and efficient. Therefore, it has been created in C# where memory automatically manages itself due to the nature of the language thus keeping it optimal	M	Y
Usability	Each key press in the game will relate to an action performed by either the player or the AI	M	Y
Response Time	The average response time between key press and reaction must be less than 0.5 seconds to ensure responsiveness	S	Y
Required Resources	The game must be able to operate on a system which has a minimum of 2GB of RAM and it must take up less than 1GB of storage space	S	Y
Frame Rate	The game is set to achieve no less than 30 frames per second as long as the computer in question can graphically support it	S	Y
Platform	The game must be able to run on any Windows based system with a requirement of Windows 7 or above	S	Y

11.2 Requirements Accomplished

In general, the majority of the requirements have been achieved which is an extremely satisfying result, but unfortunately not to the initially intended extent. From the beginning this project always had the potential to be extremely difficult and in order to manage it certain shortcuts had to be taken with the hopes of achieving a Minimum Viable Product within the available time. Proof of concept was always the aim but being unable to achieve full autonomy with the NPC was disappointing. But as a whole the project can be deemed as a success because the most important requirements have all been met to some extent.

11.3 Neglected Considerations

Some of the requirements that were of a lower priority on the list which were met, were not achieved at the highest possible level due to time constraints and this is something that was overlooked for majority of the development stage and can only now be reflected upon properly. As mentioned previously, the biggest downfall of this project is the level of autonomy that the NPC managed to achieve, due to a lack of knowledge because the subject was entirely new but mostly due to time.

Something that was not even considered until very late was cross-platform compatibility. The executable application in its current state can only be run through a Windows based computer unless opened through Unity which is the development software. In terms of a proof of concept this is not something that even needed to be considered but initially the end goal was to create not only a proof of concept, but a game.

11.4 Further Development

As mentioned toward the end of the previous paragraph, the biggest future development possible would be to turn this into a marketable game which can be put online for thousands of people to enjoy and test. Of course, this would involve multiple changes to bring the prototype in its current state to that point. The first of which being to add several more complex behaviours to the Finite State Machine to give the NPC more depth and a larger sense of intelligence, bringing it one step closer to full autonomy. Although a game like this would never normally even try to achieve that level of autonomy, pushing the limitations is how progress is made in terms of new developments.

The next thing which could be improved is the game environment. Currently, it is already aesthetically pleasing but repetitive and if this prototype were to become a real-world game it would need to be more interesting with multiple levels which all had its own unique scenery. The concept of this is easily achievable but it's something that requires a lot of time and special attention to detail which was not a luxury awarded for the duration of the project.

12 Conclusion

12.1 Evaluation of Report

In the beginning stages of this project the entire process was made clear in the form of a Gantt chart so that it could all be planned out accurately in accordance of what needs to be done first. Although the proposal is the first piece of evidence toward this project that was submitted, once it had been completed focus was drawn directly to the implementation stage and this was a mistake because, although the implementation went relatively well in terms of what was achieved, it had the potential to be so much better if the requirements had all been planned prior to even beginning the first prototype because a clear path would have been laid to follow.

The usual setbacks occurred as it would with any project, but unprecedented circumstances to do with worldly affairs also played a large role in hindering the potential of the report in general because the majority of development had already been completed by this stage but the report was ongoing until the final moments before submission and with everything happening there were some negative impacts.

But the relevant research topics to do with this project were all covered to some level of depth and no crucial information is missing, but had time not been an obstacle even further research could have been done toward the core of this project which is the behavioural tendencies behind the AI NPC, which could have aided in creating a greater understanding, which would have a domino effect and possibly even speed up development.

12.2 Evaluation of Product

Overall the project was deemed as a success because all of the requirements were met, but to varying extents and that is the largest overall criticism. Had there been more time then there would not have been the need to revise certain requirements which were made early on and resulted in being practically unachievable because of the knowledge required to implement them but mostly due to time and outside interferences which have played a large role in the hindrance of the completion of the report and implementation to some degree.

Therefore it cannot be said that the final product as a whole was a complete and utter success because there are many things that were not done or achieved to the highest possible standard. But as a whole, given current circumstances, what was achieved is credible and still required a lot of time and effort, but there is still some level of disappointment.

12.3 Evaluation of Personal Performance

Although a daunting task, the opportunity to commit so much time and effort into one final piece of work which represents an individual's skillsets and capabilities was genuinely an enjoyable task with many learning outcomes which I cannot even begin to describe in detail, which go beyond this project and into my personal life.

I chose this particular topic because it has been my dream for over a decade to work within the world of video games because they have had a large impact on who I am, and because of my large interest in artificial intelligence. I would love to create something that can bring joy to people around the planet.
I gained many new skills over the duration of this year because I pushed myself to at least try and achieve a good result at the end of it all. I began the year extremely motivated to produce something admirable and I believe that my initial efforts reflected this but as time went on and I got into the depth of the research toward this report, I started to realise that I had potentially bitten off more than I could chew. These doubts were reaffirmed when the development stage began because although I have a lot of experience in programming with the C# language, video game development presents an entire new way of writing this language because of the new syntaxes and video game logic.

With these doubts in mind I had to revise the initial requirements set because they were just too difficult to achieve with the given time constraints, and even with the revised requirements I encountered some issues which would halt development for weeks at a time but thankfully I was surrounded by knowledgeable friends and a great supervisor who could always assist with the bug fixes which slowed development. Even with these positive factors, productivity and progress came to an absolute halt around early March when an extremely close family member passed away just as the Covid19 pandemic was beginning and it had a large emotional impact on me which had me distracted for weeks.

Many of my family members from all around the world came to be with us during this difficult time for the funeral and it made things easier from an emotional perspective, but I later realised that it was just another distraction which was keeping me from focusing on my work and although it is honestly what I needed to make it through the situation, I also knew that I was losing too much time regarding the completion of this project and I suffered severe anxiety as a result. I shortly contacted the head of faculty at the university explaining the situation because I believed that there was no possibility that I could meet the deadline at this point with all of the lost time, and so I asked for her advice.

Her response was a saving grace because she told me that I was completely eligible to apply for extenuating circumstances with my current situation, but she also told me that the university was planning on giving all final year students a 3-week extension for the project because of the state of the world due to the ongoing pandemic which is affecting everyone to some degree. And with this information in mind, I set a goal that I would put all of my focus and energy into the completion of this project to a sufficient standard, and I believe that I achieved this.

And so I would like to conclude by saying that although many challenges were faced throughout the duration of this project, I am satisfied with the things I have learnt and with the overall product that I achieved because it is without a doubt going to aid me in my future career plans. And at the end of the day what matters more than the grade that is given for the project as a whole is what I can take away from this experience with the learning outcomes and how they can and will be applied to my life. Despite everything, I did put in a lot of effort to produce this and whilst that is a good thing, the project as a whole experience was rather sobering at times when I realised my capabilities with my current knowledge, and it has driven me to strive to learn more in all aspects of life.

13 Bibliography

CSharp-Station Team. (2018). *Understanding the Differences Between C#, C++, and C.*

Day, T. (2013). *Success in Academic Writing*. Palgrave.

Portocarrero, J. M., Delicato, F. C., Pires, P. F., Gámez, N., Fuentes, L., Ludovino, D., & Ferreira, P. (2014). Autonomic Wireless Sensor Networks: A Systematic Literature Review. *Journal of Sensors*.

Rudestam, K. E., & Newton, R. R. (2007). *Surviving Your Dissertation: A Comprehensive Guide to Content and Process* (3rd ed.). SAGE.

Šilingas, D., & Butleris, R. (2015). Towards implementing a framework for modeling software requirements in MagicDraw UML. *Information Technology and Control, 38*(2).

14 Appendices

14.1 Appendix A – Proposal

1 Overview

The intention of this project is to successfully develop and fully implement an artificial intelligence in the form of an NPC (Non-Player Character) which possesses the quality of self-sufficient behaviour in a 2-Dimensional environment which will be showcased as a 2D game created in the Unity engine. This differs from a basic artificial intelligence which expresses no self-sufficiency in decision making characteristics. These characteristics can be broken down into three subcategories which truly define a complex AI from something much simpler that doesn't possess behavioural tendencies. Sense, the ability for the agent to detect (or is told about) things in their environment that may influence their behaviour. Think, the agent then decides on how it should respond to the thing it detected. Act, the agent performs an action which puts the previously made decision into motion. And now the three-step cycle is complete, and the situation has changed, therefore the cycle can be repeated with new data (Sizer, 2018). The game being created for this project shall consist of an

advanced AI which has the ability to patrol within its construct without being given a path that it must use. It shall also possess the ability to decide when it should transition from the patrolling state to the aggro state which will allow it to chase the player and try to inflict enough damage upon the player, thus ending the game for the player. This top down combat style type of game is still very popular to this day as similar games have been released in the past few years and have found extreme success bringing in millions of pounds. Concluding that with the right interpretation and approach, 2D games can be still extremely relevant in this day and age although majority of game developers have moved toward the 3D approach. This has had no bearing on the success of a well implemented 2D game which executes a great concept. The inspiration for this project came from the indie developer known as Edmund McMillen and his game 'The Binding of Isaac', which possesses similar characteristics to that of this project. Edmund became an overnight success with the release of 'Super Meat Boy', his first title, deemed "one of the greatest games ever created", followed by 'The Binding of Isaac' which had similar success (Glagowski, 2018).

The game will be created in the Unity Engine as a 2D project with the backend being C# and once implemented the project will be converted to a WebGL build which allows a project to be published on the web as a JavaScript program with the use of HTML5 technologies and the WebGL rendering API to run Unity content in a web browser, thus allowing the project to be accessible from a website that shall accompany the game, making for simplified demonstration across any desktop platform which possesses a browser. To put it simply, the final product will be a Roguelike 2D Top-Down Dungeon Crawler style combat game.

Keywords: Artificial Intelligence, 2-Dimensional Environment, Behaviour, Self-sufficiency, Sense, Think, Act, Dungeon Crawler, Roguelike, Unity Engine, C# (Programming Language).

2 Aim

The aim of this project is to successfully investigate into the possibility of fully implementing a 2-Dimensional game which contains an advanced AI in the form of a Non-Player Character. This NPC shall possess the ability of self-sufficient behavioural tendencies which will allow it to sense, think and act based on the way it interacts with the player as this will influence the decisions it makes, and input it's receiving from the environment. It will be showcased as a game developed in the Unity engine with a C# backend.

3 Objectives

- Research Report / Total = 8.0 days

A detailed and specific report containing in-depth information and research regarding the various elements of the chosen topic which shall allow for the completion of this project.

1. Initial proposal [2.0]
- First document submitted presenting the initial project idea along with justification

2. Literature Review [3.0]
- Citing other people's literature and reviewing it to build a supporting case

3. Final Research Aspect [3.0]
- All research components brought together and combined for the final draft

- Design Documentation / Total = 4.0 days

This section of the project specifies the technologies used and the system architectures which shall be put in place during the implementation phase.

4. Requirement Analysis [1.0]

- Identifying the high-level requirements which will be needed for the system

5. Statement of Requirements (Requirement Specification) [1.0]
- Listing the actual high-level requirements identified during the previous stage

6. Paper Prototype [2.0]
- Designing a prototype of the game but on paper to help visualise the idea

- Implementation / Total = 35.0 days

At this point the Paper Prototype has been fully realised and implemented, fulfilling the requirement specification which was created during the previous phase of the project.

7. Base functionalities implemented/Realised Prototype [Total = 7.0 days]

- Paper prototype being adapted and reimagined as a real system [2]

- Base environment created within Unity [2]

- Base environment updated to become much more detailed and ready for the AI [3]

8. Full Implementation of Requirements [Total = 21.0 days]

- Basic Requirement Specs implemented [4]
- Basic Artificial Intelligence implemented [5]
- Artificial Intelligence improved to have more complex behaviour [8]
- Optimising the code to use less memory [2]
- Additional features added if there is ample time [2]

9. Blackbox/Whitebox Testing [Total = 7.0 days]

- Analysing and evaluating any immediate bugs found [2]
- White Box: Testing the internal structure and implementation myself [4]
- Black Box: Allowing an outside tester to examine the internal structure and implementation [1]

- <u>Evaluation Report</u> / Total = 2.0 days

A detailed report evaluating the overall outcome of the final product which should contain a reflection of what could've been done to improve the result.

10. Evaluation of Report [0.5]

- Evaluating the quality of the completed report

11. Evaluation of Product [1.0]

- Evaluating the quality of the overall product and the achieved outcomes

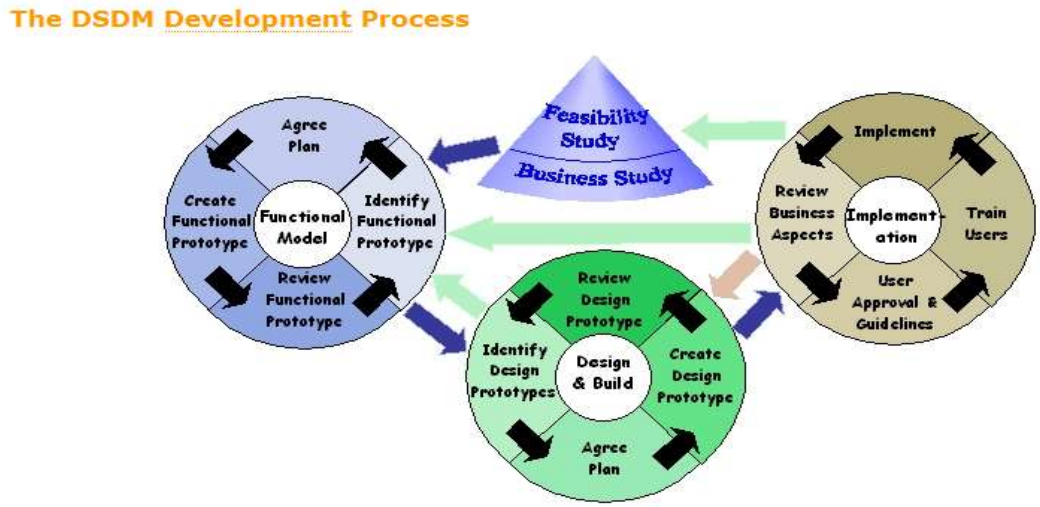
12. Evaluation of Personal Performance [0.5]

-Evaluating the work ethic and goals achieved over the entire duration of the project

3.5 Project Framework or Any Methodology used

When taking into consideration the vast number of methodologies and frameworks available for use, when referring to the subject of game development the most understandable approach would be the Dynamic Systems Development Method (DSDM). The reason for this is because of the flexibility that DSDM offers. Something like the Waterfall model could never be applied to a situation like this because there needs to be constant iterations of development and testing which repeat. When implementing this methodology game development begins with the creation of basic features which places the game at a useable and marketable state, and if not, at the very least reusable so that content never goes to waste regardless of whether or not the full requirement specification was met. The process of starting off with a basic design and progressively moving toward complexity is the organic flow that DSDM promotes.

The nature of the methodology allows for this as things are done step by step in an iterative motion thus allowing for the most important thing to always be implemented next in a high-level specification prioritisation. Building incrementally from firm foundations is the core of this approach. Finding the fun is a massive part of game development because there needs to be interesting features which attract the users, and with this methodology the users are actively involved during the development of the system thus making them more likely to embrace it and at the same time the constant feedback ensures that the system meets the requirements. With the implementation of this approach the system is more than likely to deliver the agreed specifications because of the type of workflow that coincides with this developmental methodology.



4 Legal, Social, Ethical and Professional

In terms of legal issues associated with this project, the key factor that needs to be taken into consideration is Intellectual Property Rights (IPR). Stealing another individual's idea which has already been created can breach copyright law as it's theft of intellectual property. In this scenario there is inspiration from other games but no direct plagiarism and/or theft of intellectual property. If the game were to consist of gambling aspects which involved real currency, then it may also be in breach of the law depending on the audience it's set out for. There is a fine line between social gaming and gambling which may not be crossed, but for this project there is no use of in-game or real-world currency. The system of formal self-regulation in this industry is known as PEGI (Pan-European Game Information), and it ensures that the targeted audiences are kept safe offline and online by setting appropriate age restrictions. PEGI has a guideline which needs to be followed and kept in accordance with, in regard to advertising as well, based on age restrictions certain games may only be advertised in a certain way and on specific platforms. In the context of this project all relative laws will be abided by.

Video games can have a meaningful social impact but in this case the design and implementation are relatively simple compared to existing Triple-A game titles which may dramatically influence the way a person thinks, thus preventing the possibility of addiction which can lead to social ineptness.

Interpersonal abilities may be impacted as an individual can spend so much time in a game that they lose touch from reality and begin living a secluded lifestyle. In regard to this project these described circumstances will most likely not be an issue as the end product is a game which can be completed in a mere matter of minutes.

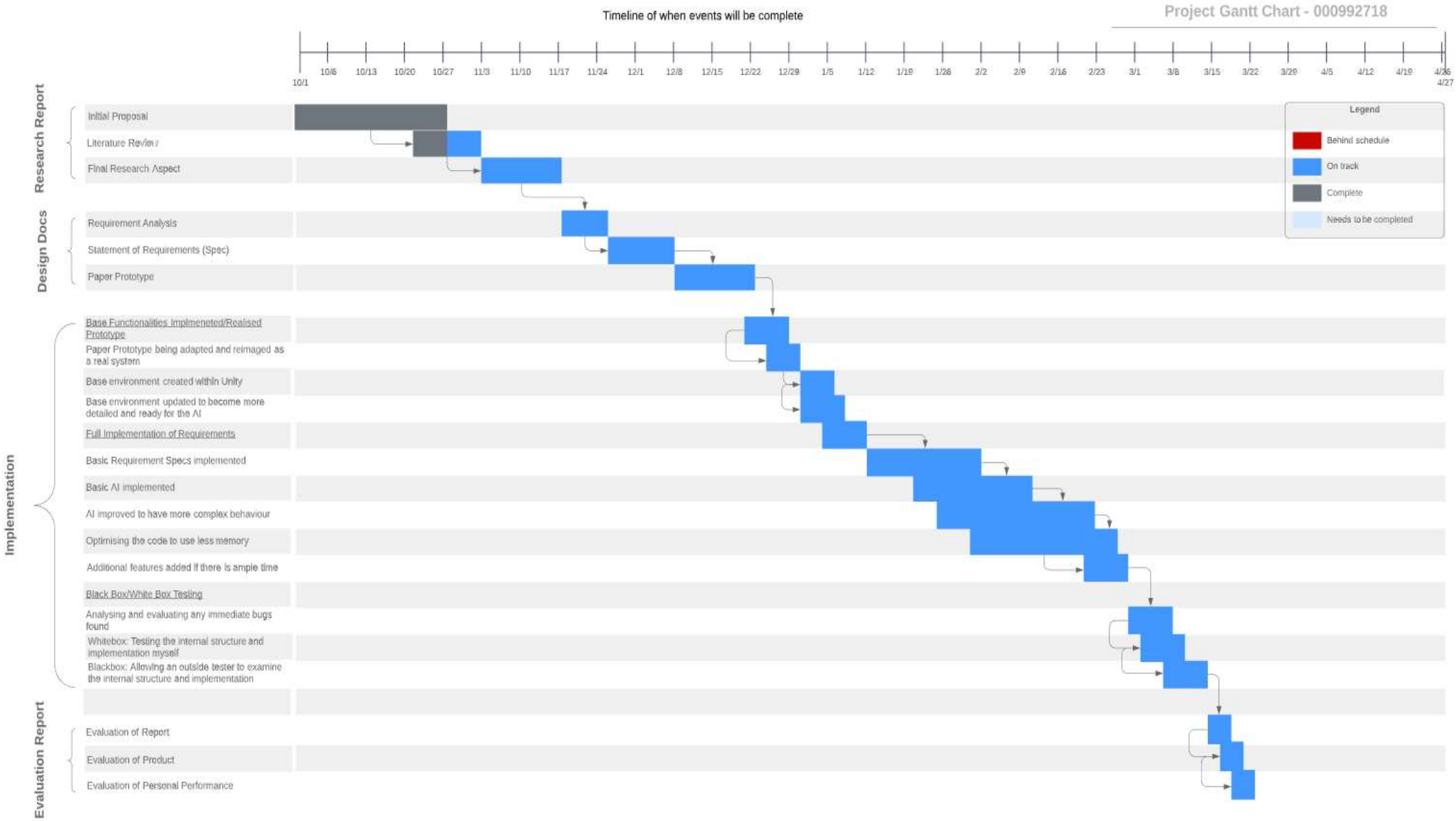
Ethical issues in regard to the video game industry is a massive topic which can be summarised in a few concise points and although it's relative to all games no matter the genre, it's very much so based on the type of content existing within the game. The key issues are with violence, sex, rating, stereotyping, addiction within the community, and bad social habits such as smoking, drinking or even the consumption of drugs. The major concerns with the aforementioned topics are that they desensitise players and can change their perception of what they believe is acceptable and this leads to a generation of individuals potentially making questionable life decisions based on the influence of a game. In regard to this project, because of the simplicity of the overall concept and design these factors mentioned above will not play a role. The only issue that may apply is a minor display of violence because in the 2D game there will be Non-Player Characters (NPCs) which chase after and attack the player to prevent him/her from achieving the end goal which is navigating to the final level and defeating the boss which will commence the end of the game. No graphic displays of violence will occur, and the overall artwork style is retro and pixelated which are child friendly.

The professional issues are very much so integrated with the ethical issues in regard to this industry which have been covered in detail above. One subject worth mentioning is "The Independent Games Developers Association (IGDA) which is an organisation that supports people making their games independently, they have been assisting all types of game developers from coding to script writers. Their mission is to advance the careers and enhance the lives of game developers by connecting members with their peers, promoting professional development and advocating on issues that affect the developer community." - (Edwards, 2012)

This article is of interest because it demonstrates the type of professional bodies that exist within the industry that support indie (solo or part of a small team) developers and they may be of great use in assisting indie developers in establishing a footprint in the world of game development. If this project is a success, then an organisation such as the one mentioned may be of great use.

5 Planning (see appendix A)

Please see the diagram on the next page:


Timeline of when events will be complete
Project Gantt Chart - 000992718
10/1
10/6
10/13
10/20
10/27
11/3
11/10
11/17
11/24
12/1
12/8
12/15
12/22
12/29
1/5
1/12
1/19
1/26
2/2
2/9
2/16
2/23
3/1
3/8
3/15
3/22
3/29
4/5
4/12
4/19
4/26
4/27
Research Report
Initial Proposal
Literature Review
Final Research Aspect
Design Docs
Requirement Analysis
Statement of Requirements (Spec)
Paper Prototype
Implementation
Base Functionalities Implemeneted/Realised Prototype
Paper Prototype being adapted and reimaged as a real system
Base environment created within Unity
Base environment updated to become more detailed and ready for the AI
Full Implementation of Requirements
Basic Requirement Specs implemented
Basic AI implemented
AI improved to have more complex behaviour
Optimising the code to use less memory
Additional features added if there is ample time
Black Box/White Box Testing
Analysing and evaluating any immediate bugs found
Whitebox: Testing the internal structure and implementation myself
Blackbox: Allowing an outside tester to examine the internal structure and implementation
Evaluation Report
Evaluation of Report
Evaluation of Product
Evaluation of Personal Performance
Legend
Behind schedule
On track
Complete
Needs to be completed

6 Initial References

1. Stapleton, J. (1999). DSDM: Dynamic Systems Development Method - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/779095 [Accessed 11 Oct. 2019].

2. Richards, K. (2004). What is DSDM and the 8 principles. [online] Agilekrc.com. Available at: https://agilekrc.com/printpdf/168 [Accessed 11 Oct. 2019].

3. Abushama, H. and Mohammed Hamed, A. (2013). Popular agile approaches in software development: Review and analysis - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/6633925 [Accessed 11 Oct. 2019].

4. Halberstam, S. (2017). Video Games and the Jurassic future - the legal issues | Weblaw. [online] Weblaw. Available at: https://www.weblaw.co.uk/articles/video-games-and-the-jurassic-future-the-legal-issues/ [Accessed 11 Oct. 2019].

5. Conrad, B. (2010). Six Problems Caused by Video Game Addiction - TechAddiction. [online] Techaddiction.ca. Available at: http://www.techaddiction.ca/addiction-to-video-games.html [Accessed 11 Oct. 2019].

6. Hubeek, L. and Schillemans, W. (2012). Video Games: Ethical Issues faced by Producers and Consumers. [online] Masters of Media. Available at: https://mastersofmedia.hum.uva.nl/blog/2014/09/11/video-games-ethical-issues-faced-by-producers-and-consumers/ [Accessed 11 Oct. 2019].

7. Sizer, B. (2018). The Total Beginner's Guide to Game AI. [online] GameDev.net. Available at: https://www.gamedev.net/articles/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/ [Accessed 19 Oct. 2019].

8. Glagowski, P. (2018). Life after death: Edmund McMillen on the success of The Binding of Isaac and his future. [online] Destructoid. Available at: https://www.destructoid.com/life-after-death-edmund-mcmillen-on-the-success-of-the-binding-of-isaac-and-his-future-523209.phtml [Accessed 19 Oct. 2019].

9. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 1). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1 [Accessed 27 Oct. 2019].

10. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 2). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-2/ [Accessed 27 Oct. 2019].

11. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 3). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-3/ [Accessed 27 Oct. 2019].

12. Chaudhari, K. (2018). AI for Unity games: Emulate real-world senses in NPC agent behavior. [online] Packt Hub. Available at: https://hub.packtpub.com/ai-unity-game-developers-emulate-real-world-senses/ [Accessed 27 Oct. 2019].

13. Lou, H. (2017). AI in Video Games: Toward a More Intelligent Game - Science in the News. [online] Science in the News. Available at: http://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/ [Accessed 27 Oct. 2019].

14.2 Appendix B – White Box Testing Methods and Evidence

Function Tested:	Input:	Code:	Expected Result:	Actual Result:

Player mobility	W,
	A,
	S,
	D
		public class PlayerController : MonoBehaviour
		
		{
		public float speed;
		Rigidbody2D Player;
		public Vector2 MovementVelocity;
		
		void Start()
		{
		Player = GetComponent<Rigidbody2D>(); //specifying which rigid body the RigBod variable is equal to
		}
		
		// Update is called once per frame
		void Update()
		{
		if(Input.GetKeyDown(KeyCode.Escape))
		{
		SceneManager.LoadScene(0); //escape key brings user back to main menu
		}
		
		Vector2 MovementInput = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical")); //Raw function added to create snappy effect by responding to key touches
		
		MovementVelocity = MovementInput.normalized * speed;
		}
		
		void FixedUpdate() //all code related to adjusting in game physics falls within this method
		{
		Player.MovePosition(Player.position + MovementVelocity * Time.fixedDeltaTime); //every physics step is being processed while the game runs as long as it receives input
		}
		
		}
			Player is capable of all expected functionality to do with mobility within the environment and performs it flawlessly.	Player is capable of all expected functionality to do with mobility within the environment and performs it flawlessly.

Player ranged attack	Left arrow key,
	Right arrow key,
	Up arrow key,
	Down arrow key
		public class PlayerController : MonoBehaviour
		
		{
		public float speed;
		Rigidbody2D Player;
		public Vector2 MovementVelocity;
		public float SnotShotSpeed;
		public float SnotShotDelay;
		private float LastFired;
		
		void Start()
		{
		Player = GetComponent<Rigidbody2D>(); //specifying which rigid body the RigBod variable is equal to
		}
		
		// Update is called once per frame
		void Update()
		{
		if(Input.GetKeyDown(KeyCode.Escape))
		{
		SceneManager.LoadScene(0); //escape key brings user back to main menu
		}
		float SnotShotHor = Input.GetAxis("ShootHorizontal");
		float SnotShotVer = Input.GetAxis("ShootVertical");
			Player is capable of launching projectile ranged attacks at the enemy and inflicting damage to wound the NPC.	Player is capable of launching projectile ranged attacks at the enemy, but a mistake was made early on in the project to do with physics which prevented the player from actually inflicting damage upon the NPC but regardless,

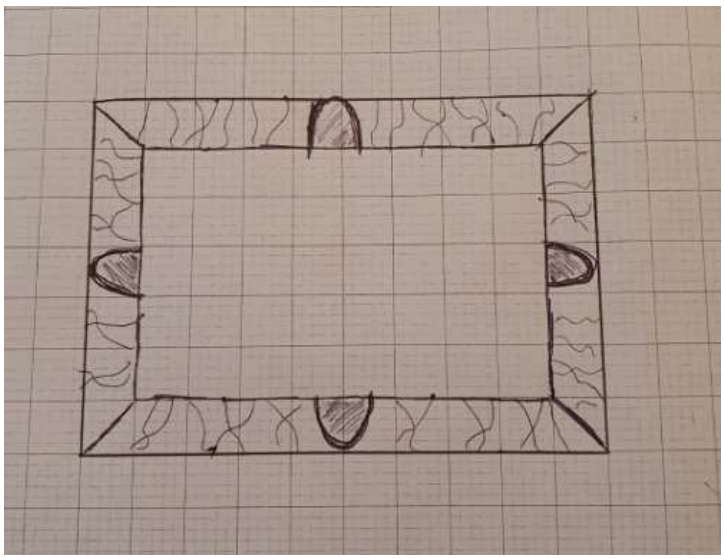
		if((SnotShotHor != 0 || SnotShotVer != 0) && Time.time > LastFired + SnotShotDelay)
		{
		SnotShoot(SnotShotHor, SnotShotVer);
		LastFired = Time.time;
		}
		
		Vector2 MovementInput = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));
		
		MovementVelocity = MovementInput.normalized * speed;
		}
		
		void FixedUpdate()
		{
		Player.MovePosition(Player.position + MovementVelocity * Time.fixedDeltaTime);
		}
		
		public void SnotShoot(float x, float y)
		{
		GameObject Tear = Instantiate(SnotShot, transform.position, transform.rotation) as GameObject;
		Tear.AddComponent<Rigidbody2D>().gravityScale = 0;
		Tear.GetComponent<Rigidbody2D>().velocity = new Vector3(
		(x < 0) ? Mathf.Floor(x) * SnotShotSpeed : Mathf.Ceil(x) * SnotShotSpeed,
		//ternary operator which acts as a boolean only checking for a true or false value
		(y < 0) ? Mathf.Floor(y) * SnotShotSpeed : Mathf.Ceil(y) * SnotShotSpeed,
		0
		);
		}
		
		}
				ranged attacks can still be launched toward it.

NPC Wander	Finite State Machine Boolean	public class PatrolBehaviour : StateMachineBehaviour
		{
		private int RandomLocation;
		public float speed;
		private LocationsPatrolled patrol;
		
		override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
		{
		patrol = GameObject.FindGameObjectWithTag("LocationsPatrolled").GetComponent<LocationsPatrolled>();
		
		RandomLocation = Random.Range(0, patrol.PointsOfPatrol.Length);
		
		}
		
		override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
		{
		if (Vector2.Distance(animator.transform.position, patrol.PointsOfPatrol[RandomLocation].position) > 0.2f)
		{
		animator.transform.position = Vector2.MoveTowards(animator.transform.position, patrol.PointsOfPatrol[RandomLocation].position, speed * Time.deltaTime);
		}
		
		else
		{
		RandomLocation = Random.Range(0, patrol.PointsOfPatrol.Length);
		
		}
		
		if (Input.GetKeyDown(KeyCode.Space)) {
		animator.SetBool("isPatrolling", false);
		}
		}
			NPC is able to wander between waypoints within the environment flawlessly with no issues encountered until it is within range to chase the player.	NPC is able to wander between waypoints within the environment flawlessly with no issues encountered until it is within range to chase the player.

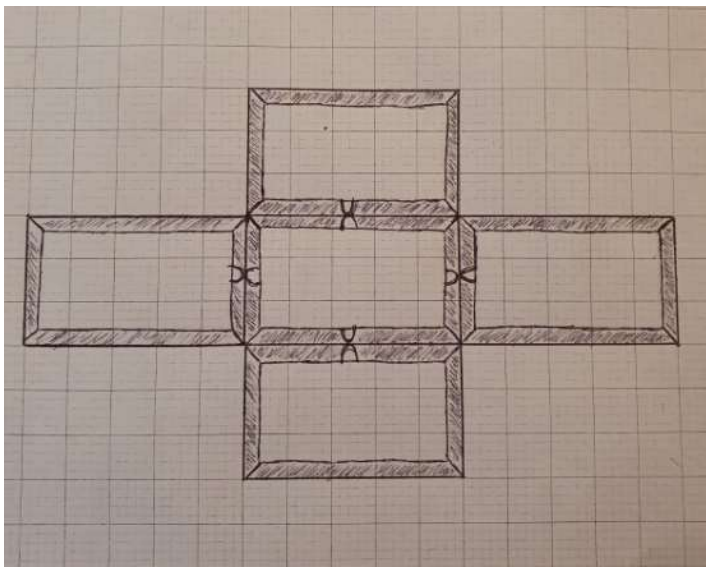
NPC Chase	Finite State Machine Boolean	public class ChaseBehaviour : StateMachineBehaviour
		{
		private Transform PositionOfPlayer;
		public float speed;
		
		// OnStateEnter is called when a transition starts and the state machine starts to evaluate this state
		override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
		{
		PositionOfPlayer = GameObject.FindGameObjectWithTag("Player").transform;
		//allowing the AI to identify the player based on its tag
		}
		
		// OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks
		override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
		{
		animator.transform.position = Vector2.MoveTowards(animator.transform.position,
			When the NPC is within a certain range of the player it is able to transition between states and begin to chase the player	When the NPC is within a certain range of the player it is able to transition between states and begin to chase the player

		PositionOfPlayer.position, speed * Time.deltaTime);
		/*transitioning the animation state based on the direction it's moving toward which is the position of the player.
		And calculaiting speed based on the speed and position of the player*/
		
		if (Input.GetKeyDown(KeyCode.Space)) //fetches the input from the spacebar keypress to use as a trigger
		{
		animator.SetBool("isChasing", false); //setting the pool parameter for the animator to false upon key press
		}
		
		if (Input.GetKeyDown(KeyCode.P))
		{
		animator.SetBool("isPatrolling", true);
		}
		}		

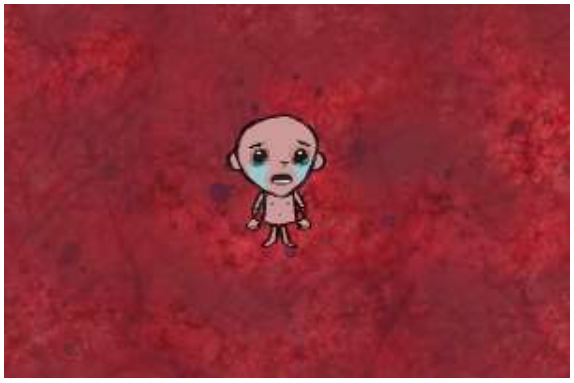
14.3 Appendix C – Initial Paper Prototype 1



14.4 Appendix D – Initial Paper Prototype 2

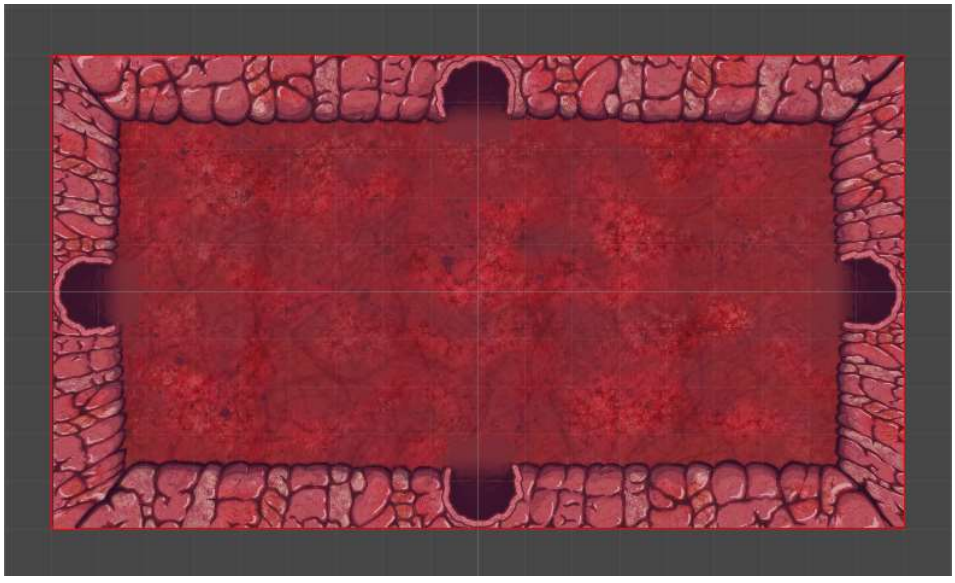


14.5 Appendix E – Screenshot of Player

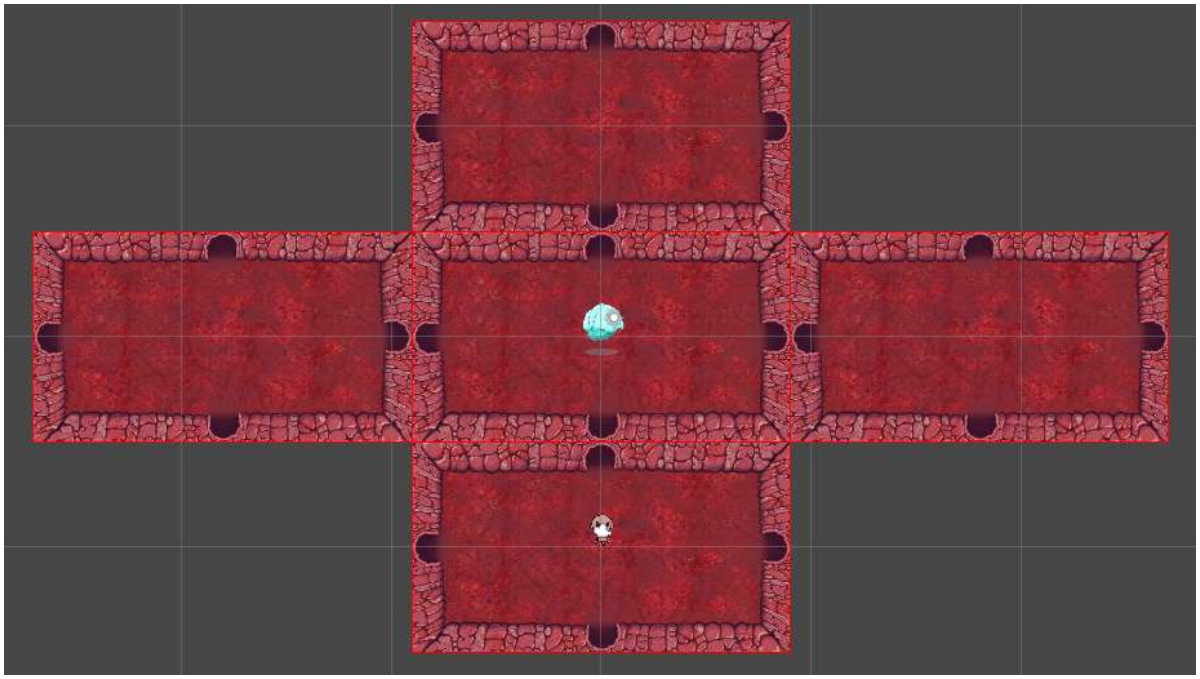


14.6 Appendix F – Screenshot of NPC

14.7 Appendix G – Screenshot of Prototype 1 implemented



14.8 Appendix H – Screenshot of Prototype 2 implemented



15 References:

1. Stapleton, J. (1999). DSDM: Dynamic Systems Development Method - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/779095 [Accessed 11 Oct. 2019].

2. Richards, K. (2004). What is DSDM and the 8 principles. [online] Agilekrc.com. Available at: https://agilekrc.com/printpdf/168 [Accessed 11 Oct. 2019].

3. Abushama, H. and Mohammed Hamed, A. (2013). Popular agile approaches in software development: Review and analysis - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/6633925 [Accessed 11 Oct. 2019].

4. Halberstam, S. (2017). Video Games and the Jurassic future - the legal issues | Weblaw. [online] Weblaw. Available at: https://www.weblaw.co.uk/articles/video-games-and-the-jurassic-future-the-legal-issues/ [Accessed 11 Oct. 2019].

5. Conrad, B. (2010). Six Problems Caused by Video Game Addiction - TechAddiction. [online] Techaddiction.ca. Available at: http://www.techaddiction.ca/addiction-to-video-games.html [Accessed 11 Oct. 2019].

6. Hubeek, L. and Schillemans, W. (2012). Video Games: Ethical Issues faced by Producers and Consumers. [online] Masters of Media. Available at: https://mastersofmedia.hum.uva.nl/blog/2014/09/11/video-games-ethical-issues-faced-by-producers-and-consumers/ [Accessed 11 Oct. 2019].

7. Sizer, B. (2018). The Total Beginner's Guide to Game AI. [online] GameDev.net. Available at: https://www.gamedev.net/articles/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/ [Accessed 19 Oct. 2019].

8. Glagowski, P. (2018). Life after death: Edmund McMillen on the success of The Binding of Isaac and his future. [online] Destructoid. Available at: https://www.destructoid.com/life-after-death-edmund-mcmillen-on-the-success-of-the-binding-of-isaac-and-his-future-523209.phtml [Accessed 19 Oct. 2019].

9. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 1). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1 [Accessed 27 Oct. 2019].

10. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 2). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-2/ [Accessed 27 Oct. 2019].

11. Kehoe, D. (2009). Designing Artificial Intelligence for Games (Part 3). [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-3/ [Accessed 27 Oct. 2019].

12. Chaudhari, K. (2018). AI for Unity games: Emulate real-world senses in NPC agent behavior. [online] Packt Hub. Available at: https://hub.packtpub.com/ai-unity-game-developers-emulate-real-world-senses/ [Accessed 27 Oct. 2019].

13. Lou, H. (2017). AI in Video Games: Toward a More Intelligent Game - Science in the News. [online] Science in the News. Available at: http://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/ [Accessed 27 Oct. 2019].

14. CSharp-Station Team (2018). Understanding the Differences Between C#, C++, and C - C# Station. [online] C# Station. Available at: https://csharp-station.com/understanding-the-differences-between-c-c-and-c/ [Accessed 7 Dec. 2019].

15. Westera, W., Prada, R., Mascarenhas, S. et al. Artificial intelligence moving serious gaming: Presenting reusable game AI components. Educ Inf Technol (2019) doi:10.1007/s10639-019-09968-2 [Accessed 10 Dec. 2019]

16. Statt, N. and Togelius, J. (2019). How artificial intelligence will revolutionize the way video games are developed and played. [online] The Verge. Available at: https://www.theverge.com/2019/3/6/18222203/video-game-ai-future-procedural-generation-deep-learning [Accessed 10 Dec. 2019].

17. Brownlee, J. (2002). Finite State Machines (FSM). [online] Ai-depot.com. Available at: http://ai-depot.com/FiniteStateMachines/FSM-Background.html [Accessed 11 Dec. 2019].

18. Gill, S. (2004). Visual Finite State Machine AI Systems. [online] Gamasutra.com. Available at: https://www.gamasutra.com/view/feature/130578/visual_finite_state_machine_ai_.php [Accessed 10 Dec. 2019].

19. Nielsen, J. (1994). Usability Engineering. San Diego: Academic Press. p. 115–148.

20. Bycer, J. (2016). 3 Failings of Procedurally Generated Game Design. [online] Gamasutra.com. Available at: https://www.gamasutra.com/blogs/JoshBycer/20160802/278327/3_Failings_of_Procedurally_Generated_Game_Design.php [Accessed 12 Dec. 2019].

21. Nielsen, J., 2019. 10 Usability Heuristics Applied To Video Games. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/usability-heuristics-applied-video-games/> [Accessed 15 February 2020].

22. J. Aerosp. Technol. Manag. vol.6 no.4 São José dos Campos Oct./Dec. 2014 Available at: https://doi.org/10.5028/jatm.v6i4.369 [Accessed 2 May 2020].