**PROJECT PROTOTYPE**

**3102GFS – Advanced Game Development**

**Assignment 1 – Research Blog**

**14th August 2017**

**Dylan Lee – s5045023**

The goal of Project Prototype is to explore an area or areas of advanced game development. There are no restrictions on to what software, program, or language the code can be written in. The prototype will be presented on the 14th of August, it must demonstrate a clear concept and approach as well as present potential for full development. This blog will outline the research, methods and processes taken during the development of the prototype elaborating on the challenges presented and how they were overcome.

**Choosing an Area to Explore**

The first thing that needed to be done was to choose an area of advanced game development to begin exploring. Research was done into many different areas of programming, guided by the links and resources provided on Learning at Griffith. This research took place over a period of three weeks in addition to the tutorial each week. After reaching the end of this research period, AI or artificial intelligence was the area of advanced game development that was chosen to be explored for this project. The goal for my project was to create an FPS (First Person Shooter) agent in *Unity 3D* that could run and complete a training or obstacle course, shooting down some targets, picking up health and ammo packs when needed.

**Session 1 - AI Pathing Methods**

Now that AI had been chosen, further research into the area was conducted to learn more and get a better understanding of how it works before beginning to write code. The first challenge that needed to be passed was getting the AI to move from one location on the map to the other whilst avoiding obstacles or walls. Initially I read into A\*pathing and studied an example project.

Into to A\*pathing: <https://www.raywenderlich.com/4946/introduction-to-a-pathfinding>

Example project and write up: <http://blog.two-cats.com/2014/06/a-star-example/>

A\* pathing seemed like a good option and it could be done in Unity 3D using an array to map out each area on the map. Research continued into A\* pathfinding and a tutorial written in C# was discovered and tried.

A\* pathfinding C# tutorial: <http://gigi.nullneuron.net/gigilabs/a-pathfinding-example-in-c/>

This tutorial didn’t seem overly helpful, even though the script created an example of A\* pathfinding in a string, the idea was to re write it and play around with it to understand and learn some of the fundamentals. This didn’t end up being the case, the script was out dated and I had difficulties trying to get it to work. The tutorial was unclear on where some pieces of the code needed to be written and there were difficulties downloading the source code.

**Session 2 – Navigation Mesh and Initial Programming**

Further research was done into other methods of pathing AI’s across 3D environments whilst avoiding obstacles. *Unity’s* NavMeshAgent and NavMesh baking was considered as an easy form of pathing the AI. Although this wasn’t programming the movement for the AI, the thought behind using this was more time could be spent on programming other aspects of the AI such as its decision making, vision, hearing, various states etc. *Unity’s* tutorial on navigation mesh was viewed and implemented into the project. After following the tutorial, the AI was at the point where it could run to any location on the map it was told to while avoiding basic obstacles such as walls.

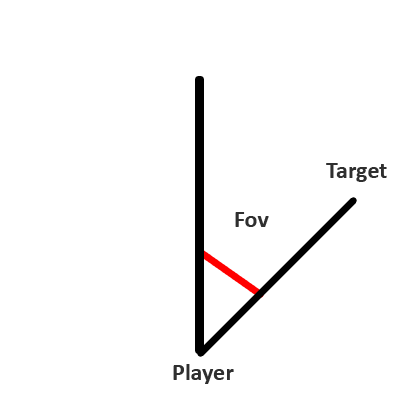
Navigation mesh tutorial: <https://unity3d.com/learn/tutorials/topics/navigation/navigation-overview?playlist=17105>

Once the navigation mesh was set up in the prototype, the next most important aspect of the AI was its ability to take down targets. To do this my AI needed to be able to see or hear targets, aim at them and fire some projectiles that would kill them. A target was set up in the game space and research was done into how to achieve AI vision. After some brief research, a Youtube tutorial was found that was helpful in guiding the start of the AI’s vision code.

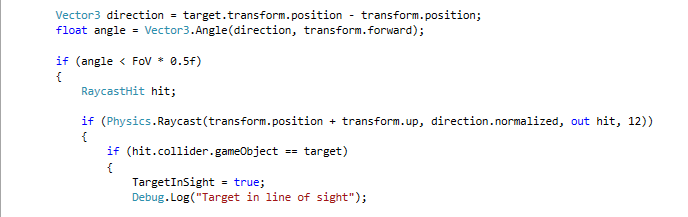
Youtube AI Tutorial: <https://www.youtube.com/watch?v=mBGUY7EUxXQ>

Pieces of the code from this video was taken and worked and adjusted for what the AI for this project needed to do. The tutorial script used an OnTriggerEnter function which for the AI wasn’t going to work. To fix this, the code was moved into the Update function.

Two different Vector 3’s were setup that start from the player, one going forwards and the other directed at the AI’s target to create a cone or angle that was then checked against a field of view (FoV) float. If the angle is less than half of the float, the AI should be able to see the target.



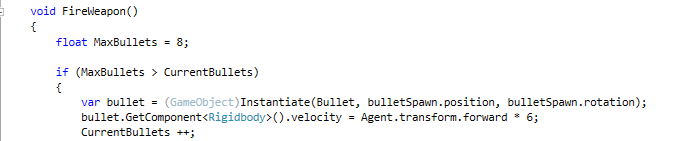
A raycast which followed the Vector aimed at the target was used to check what the AI was looking at, if the raycast hit the collider of the target, a Boolean named TargetInSight was set to true which meant the AI was looking at the target. This Boolean could then be checked to make sure the AI was looking in the right direction before it did anything else, like fire a weapon. Additionally, the AI needed to be within a certain range of the target to see it. To do this, a max distance of 20 was set to the raycast. A debug log was also implemented to tell us if this code was working before continuing.



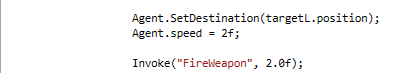
Now that the AI could see the target, the next goal was the get it to fire at the target. To get a simple and fast way of creating and firing a bullet, the *Unity* shooting (Single Player) tutorial was used and adjusted for this project.

Unity Shooting tutorial: <https://unity3d.com/learn/tutorials/temas/multiplayer-networking/shooting-single-player>

The tutorial was implemented into a FireWeapon function that would be called when the AI saw a target but there were a few problems with the script. The script was intended for a player to fire the bullets on pressing a key, so for the AI, it didn’t stop firing rapidly until the target was destroyed, spawning a very large number of bullets. To prevent this, a max number of bullets and an if statement checking the bullet count was written. If the current number of bullets exceeded the max number of bullets, the AI would stop firing.



The other major problem with how the AI fired was its accuracy, as it turned toward the target, it would fire as it entered the FoV, this meant that the AI hadn’t finished turning and the spawn point for the bullets hadn’t lined up with the target yet. To give the AI more accuracy, the FireWeapon function was Invoked on a timer of 2 seconds to give the AI time to finish turning, allowing the spawn location of the bullets to line up correctly. Additionally, when the AI spotted its target, its speed on NavMeshAgent component was accessed and reduced. This also helped with a sort of realism to the AI, real people don’t sprint and fire at their targets, neither should the AI.



At the current stage of the project, there are a few known issues and challenges that are going to arise as development continues. The first is that there will be a need for destroying large amounts of bullets, early research suggests I can do this with an array that adds all objects with a tag. Meaning that all the bullets can be tagged and then destroy all objects with that tag. The implementation of ammunition and health will be an important part of the AI when it comes to getting it to make decisions. For example, the AI might have a target in sight, but very low health, so its priority should be to go get the health before it looks to attack its target. Lastly, without thinking too far ahead, it will be important to see how the AI behaves when there is more than one target in the scene or even in its vision. These problems and challenges are the next goals for the development of the project.

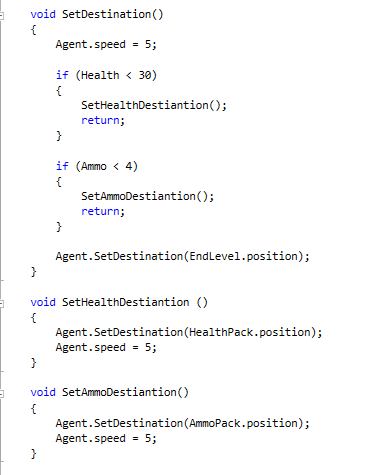
**Session 3 – Health, Ammo & Bullet Destruction**

The first goal chosen for this session was to implement a health system, to do this a float was set up as the AI’s health. A function was written that sets the AI’s destination to the location of a health pack, if the AI’s health was lower than a set value, the function was called. This worked as intended and was a good base for this stage of the project, the code and function would be utilized later in development.

Next the ammunition for the AI was set up, the same set up from the health was used for the ammunition. A float was set up for ammunition, if the amount of ammunition reached a certain value, a function was called that set the AI’s destination to an ammo pack. Additionally, when the AI fired its weapon, we subtracted 1 from the ammo float.

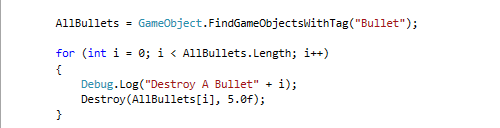
It was at this stage a problem was discovered. If the AI was left to kill the target, the destination of the AI wouldn’t change even if the conditions for either the health or ammo statements were met, it would just move towards the end of the level. The if statements work before the target is destroyed, but not afterwards. To fix this, the if conditions for the health and ammo were moved into the function that set the final destination of the AI. This meant that it would always check those conditions before the AI would head for the final destination.

This however revealed another problem, the AI would respond to the ammunition condition but not the health condition. While the script was running, if the ammo value was manually changed, the AI would react, but this wouldn’t work for the health float. This was due to a few small mistakes in our code. A return was added after we set each destination in the if statement. The health statement was listed first as this was intended to be a priority for the AI, if this condition was met, return was called so that it wouldn’t run any further, meaning the destination of the AI would not change. The same was done with the ammunition statement, then at the bottom of the function, if neither if statements were met, the destination was set to the final destination.



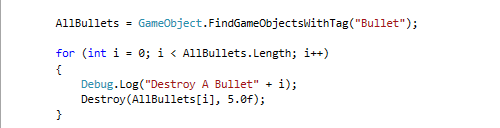
In future, the plan is to have the AI work out which is more important, ammunition or health via some sort of calculation. Forcing the AI to choose going for one over the other even if it needs both based on how much of each it’s missing.

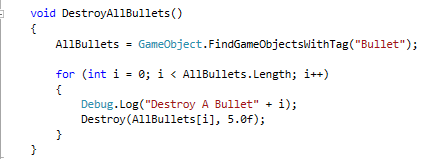
The next task undertaken was setting up an array to store our bullets so we could destroy all of them after a short duration of time. Research was done through the *Unity* Scripting API to learn how to write an array and fill it with tagged objects. An array was written and implemented into the project, and then a destroy function was called to destroy all objects in the array. The initial implementation of this did not work very well, it would destroy each bullet 5 seconds after the other instead of at the same time. This took a lot of fiddling around with different lines of code, different things such as foreach loops, for loops and while loops were written. Eventually a for loop was written that worked and destroyed all the bullets after a 5 second delay.

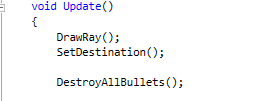


**Session 4 – Tidying Code**

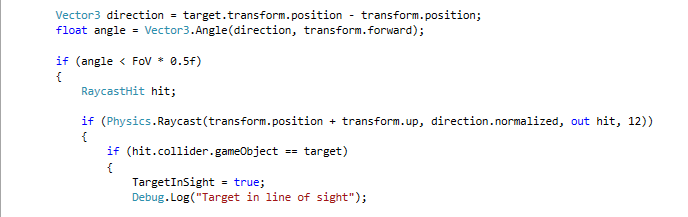
After receiving feedback during the tutorial of week 5, there were a few areas of code that could be written cleaner and at a higher level. They didn’t need to be re written but rather placed into their own functions and written there rather than in the update function. The first piece of code needing to be moved was the bullet array and the destruction of this array, fixing this was as simple as copying and pasting it into its own function then calling that function in the update.



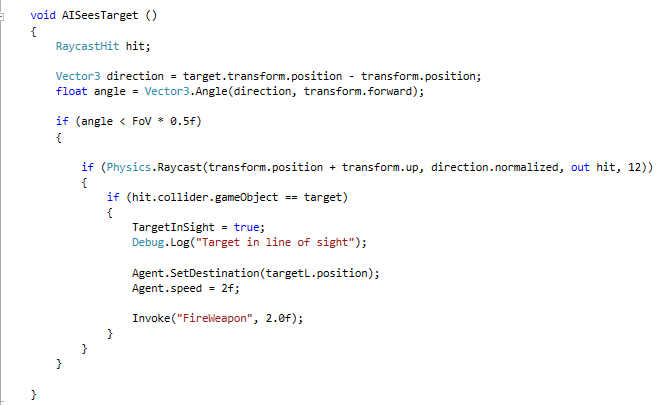


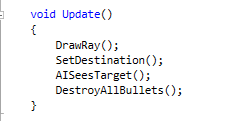


The next key area that needed to be moved out of the update function and into its own is the AI’s vision code.



Similar to the bullet array code, fixing this issue was as simple as moving all of the code into its own function then calling that function in the update.

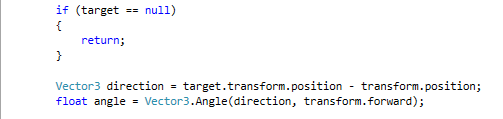




Another small issue that wasn’t causing any problems but needed to be resolved was an error in the console that appeared after the target was destroyed by the AI. The script was still trying to access the target object after it was destroyed, this was caused by one of the Vectors accessing the targets transform position.



To resolve this error, a check was added before the Vector code to see if the target is null, if it was null then the code would stop, otherwise it would continue and create the vector.



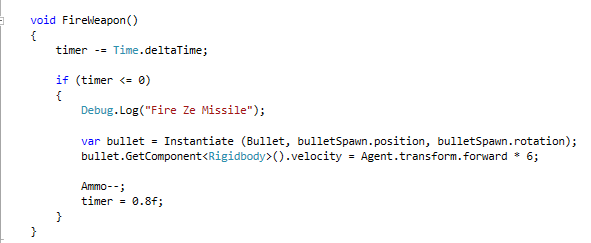
Cleaning up these multiple sections of code presented very little to no issues, moving each piece of code and checking it afterwards presented zero problems or errors with the function of the AI. Everything continued working as it did prior to the changes and the script was now written at a higher standard. Additionally, it was advised that there be two lines of spacing between functions instead of one, this was also fixed during this stage of development.

**Session 5 – Bullet Timing**

When the AI fires at his target, it fires all of its bullets instantly instead of one after the other like a real weapon. To fix this, the goal is to set up a small timer of one second and spawn a bullet at the end of the timer and restart that process over again. A float for a timer was created and set to one second within the fire weapon function. An if statement was created to check if the timer had reached 0, if it had, continue and spawn a bullet.



This first implementation didn’t work, to locate the problem, the debug logs were written into the code to see how far it was going. This revealed that there was something wrong with our timer as the “Countdown Started” showed up in our console but not “Fire Ze Missle”. This problem took a lot of moving pieces of code around and re writes that yielded no results. Eventually the problem was worked out to be a few different things. Since we had a timer for the bullets, we no longer needed to check the maximum bullets against the current bullets, so that entire if statement was removed. Secondly, the float for the timer was set to private and declared outside of the function. The AI now shot at a fire rate set by the timer float.



There was however one small issue with this set up, when we call the fire weapon function, it’s on an Invoke. Because of this, after the target is destroyed, the AI seems to fire an extra bullet or two. If the Invoke is removed, the AI fires immediately, firing its first shot as it begins to turn which was a previous issue, but it removes this new issue of the AI continuing to fire. At this stage of development this is an improvement over all and time will be spent bettering other areas of the AI before returning here.

**Future Development**

Moving into the later stages of development for this project will see the AI expand in many ways as well as its current programming become more advanced. A large goal for the AI is to achieve a hearing system, where the AI cannot see its target but is able to determine roughly where its target might be and start heading in that direction. Another key area which will see a lot of focus is the AI’s decision making and states. Right now, when the AI decided between getting ammunition, health, firing at its target and heading for the end goal, its done in a very basic and rudimentary way. Getting the AI to make more calculated and in-depth decisions in this area will be a big advancement.

Outside of these key goals, there are a few minor and more fun goals that are more of an experimentation for the project. The idea of giving the AI different weapons with different fire rates that it might use in different situations was an idea that was stumbled upon when setting the timer on the weapon fire. Different fire rates could feel like different weapons and could be incorporated into the weapons decision making and various states.