Games Development 2 – Assignment Report

# Introduction

In this report I will be discussing my implementation of the assignment and the behavior used in the implementation, this will include addition content I have added into the assignment and a discussion about different update methods.

In this assignment I have implemented everything the mark scheme has asked for as well as additional content but I did not add in formations due to time.

# Patrol & Targeting Behavior

Patrol Points Behavior

All the Patrol Behavior is mainly in the Patrol State in TankEntity.cpp but some of the waypoint implementation is in the TankAssignment.cpp.

In my implementation there are four patrol points for each time, when the scene loads up these patrol points are place at set positions but the patrol points for each team are across from each other for example: P1 – {-20,0.5,0} P2 – {20,0.5,0}), there are three buildings place so the tanks can see each other.

The implementation is designed to support up to any patrol points as it uses vectors to support this, if a tank reaches it last patrol point then it will start to move back to the first patrol point.

In this Assignment I assign each waypoint to a quad so there is a visual representation of the waypoints but in TankAssignment I allow the user to pick up these quads which in turn will move the patrol point and the tanks will react and move to the new patrol point.

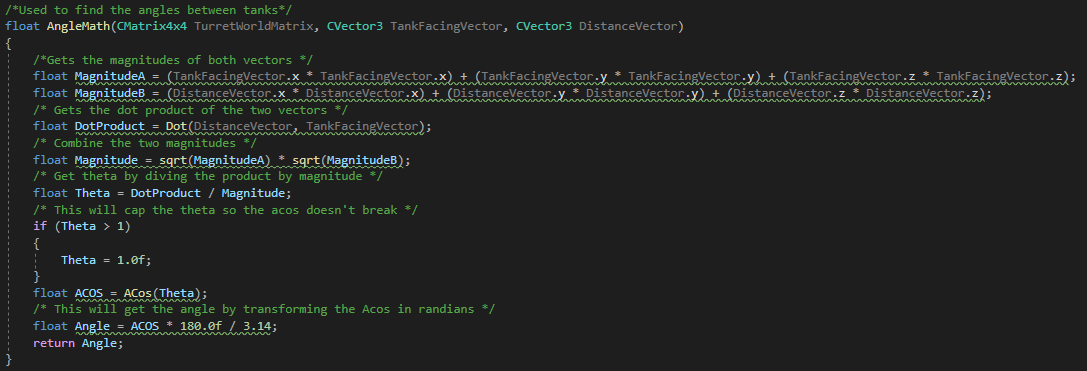
Movement Behavior

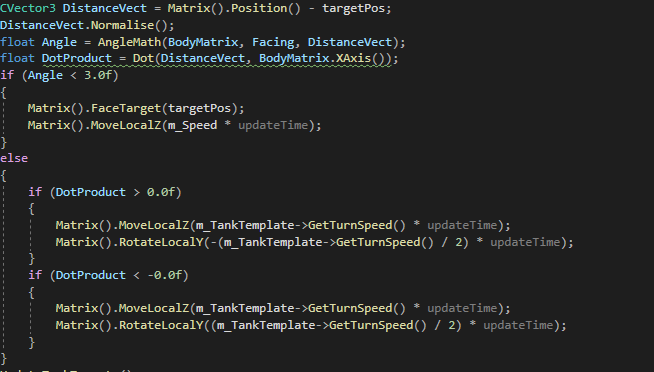
When I comes to movement in the patrol state I use a combination of angle math’s and dot product to determine if the tank need to turn left, right or move forward.

AngleMaths:

I used angle math to determine how close the rotation of an object, so in this case it’s the tank body, to get the angle to work I had to get the magnitude of the tank facing vector and the vector between the two tanks.

I then dot product of the two vectors so I can then get theta which is the dot product divided by the sqrt of both magnitudes times together.

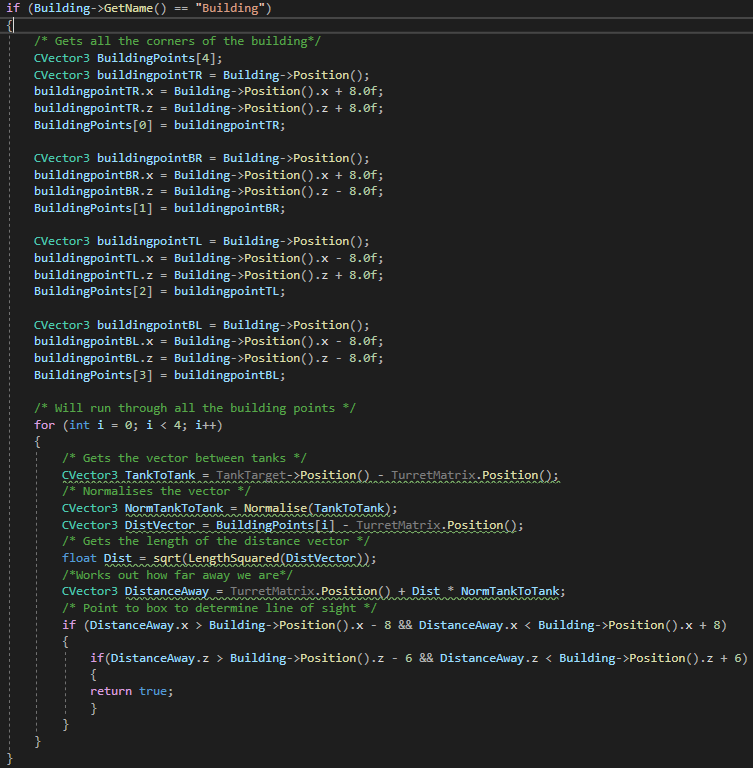
The next step is to Acos theta, but to make sure the Acos doesn’t return a invalid number I cap it so this isn’t possible, after we have the Acos we times it by 180 divided by pi and this will give us the angle.

Once we have the angle then we can just test to see if the angle is less then a certain amount. This process is also used to see if the turret angle is less then 15 which will then switch it to the Aim State.

DotProduct:

The dot product is nothing complicated, the dot product is used to see if the point is to the left or right, this will determine which way the tank will turn and what speed it needs to move.

Line Of Sight:

I used of line of sight to determine if the tank could see a another tank through a building.

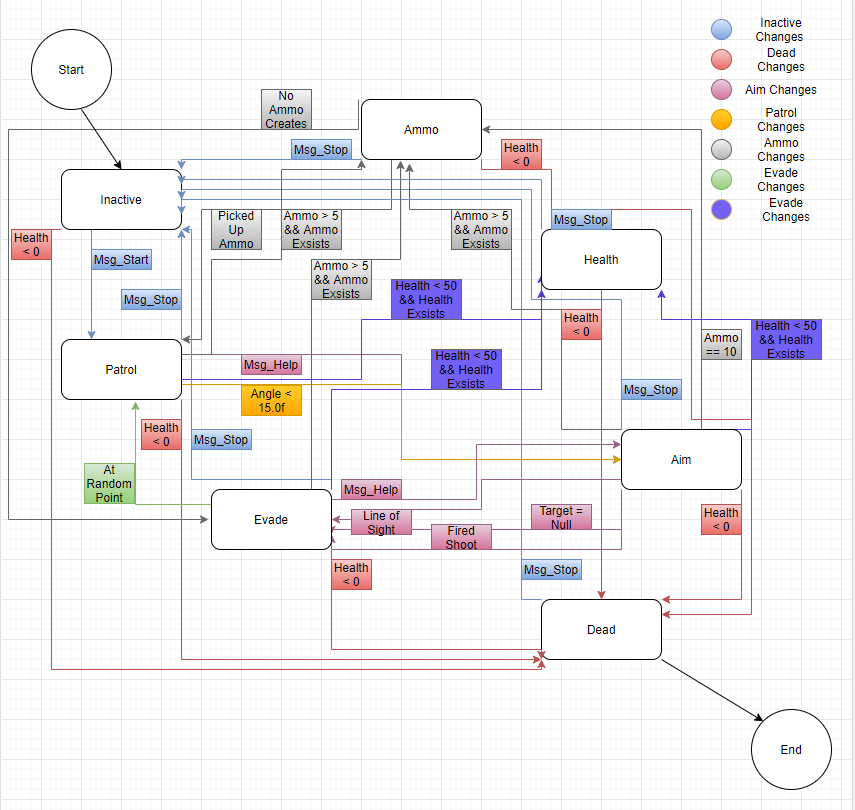
I loop through all the scenery until I get a building, I get all the corners of that building. Once I have all four points then I run a for loop on each point.

For each corner I get the distance between tanks the distance between the building and tank, add the distance of the building to tank to position and times that by the tank to tank vector and this will make tank position along the vector.

I then run a collision on it to see if its hitting any of the corners.

This is also used in the aim state to provide an accurate tank targeting system.

# State Transition Diagram



Use of States:

When it comes to using states they can be extremely useful to have, having states allows for the isolation of a segment of code, this means that if you have multiple states they won’t conflict each other unless you code them to conflict, using states in this assignment was very helpful due to the fact that each state does something different then another state and not having them might have conflicted the code.

They also are great when it comes to keep clean and readable code, if I see that the state is Patrol then I know only code related to Patrol is in there but states also allow for the connects between code to be better.

Use of Messages:

When it comes to using a messaging system they can be useful when it comes to communication between different AI entities.

I used it in the assignment to send messages across but you can also send additional information across so which made them quite useful when it came to communication between the different tanks.

Another area I used this was the Help message, this allowed me to send back the UID of the attack all the way back to the teammate tanks so they could go into the aim state and attack that target.

# Implementation of Update

SceneUpdate:

An sceneUpdate is a update function is called every frame and it will update everything inside that function, I didn’t use a sceneUpdate function in this assignment as the assignment already provided a update function to use.

But If I did implement a sceneUpdate rather then EntityUpdates then the implementation would be the same really apart from everything would have been in the same cpp file, some of the implementations of the functions might have need to change but that would be something I would just need to adept to.

Advantages:

* When you use a scene update everything is in one function call which means you don’t have to find the correct cpp.
* Debugging in a SceneUpdate would be a lot easy for the same reason as the above, everything is in the same place so you don’t have to find everything.

Disadvantages:

* Having everything in one update can affect readability quick a bit.
* The cpp can get very long if you’re updating a lot of things.

EntityUpdate:

An EntityUpdate is a update function which is broken down into lots of different update calls, this was the update function that I had in my assignment and it was super useful and was really easy to read.

Because of how the EntityUpdate is broken down you know where to find certain things that you might need instead of searching one long cpp file.

Advantages:

* Using an EntityUpdate is quite readable and easier to understand.
* The cpp’s won’t be as so its easier to find things.
* Can chose whether to update an entity or not.

Disadvantages:

* Debugging can be difficult as there are multiple updates to check and multiple c++’s.

I think that having an EntityUpdate are better then SceneUpdates as EntityUpdates are easier to read and you have a lot more control about what happens unlike in SceneUpdate where everything updates at once.

# The Basics – ALL COMPLETED

When it comes to the basic requirements of the assignment I have done everything that was asked.

# XML – ALL COMPLETED

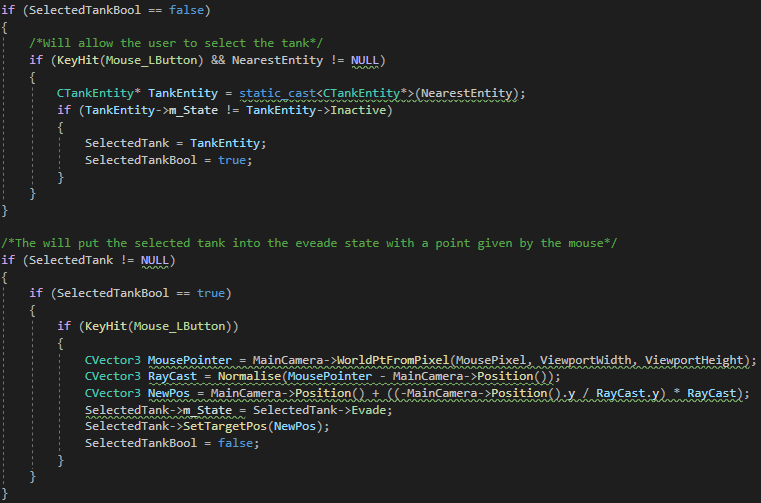
In the XML each tank has its own template, this allows for the tank to have different stats to one another, When it comes to creating the tanks in XML I had to make a different type for them, while all entities are a part of Entity then Tank are a part of Tank Entity. Because Tanks and Entities are different they need to be adapted differently.

In XML each tank has a set of patrol points so each tank can have a different patrol list, I read in 4 patrol points and assign them to a vector and when the TankEntity gets create I pass that vector with it.

The only entities to get created outside XML are the Bullets and creates but there templates are made in the XML.

# User Interface - ALL COMPLETED

When the user press the 0 key it will show some extra data about the tanks, this includes the state, ammo, HP and team. This was very simple to do as it was just rendering text next to the entities with the correct data.

When the user left clicks with the mouse it will gets the nearest entity and select it, The Tanks text will appear white if selected ,when they left click again it will put that tank into the evade state and it will move towards the position the mouse was at

When the user presses 3 it will move them to the first tank in a vector of tanks, this acts as a chase camera for the tanks, when a tank dies the chase camera will detach itself from it, if you press 3 and the first tank is dead then it will move to the next tank after that one.

# Advanced Game Behavior - 95% COMPLETED(MISSING FORMATION)

All the tanks in the assignment use the template variables that are already in the project, these variables allow me to very easily adopt it to the XML side of the project too but this also means that every tank can have different stats that are a part of the existing template.

When a tanks HP is reduced to 0 it enters the dead state, nothing can take the tank out of this dead state, in the dead state an animation will play and when that animation is finished it will delete the tank.

The tanks patrol between 4 different waypoints, but more waypoints could be added to make the patrol longer. The patrol points are attached to quads so if the quads move so do the way points, this is so if the user wants to pick up the way points they can and the AI will adapt to it.

The patrol points are between 3 building, when an AI wants to shoot a tank if the building is in the way then it won’t shoot the tank. The line of sight works like I stated above. (Read Line of Sight for more info).

In my project there are two teams of three tanks, if given more time I would have attempted to do formations but when a tank gets hit by an enemy the friendly tanks on its team will shoot the attack if they can and the states allow it.

Each tank will have ammo that is used when they fire a shell, they have a max of 10 shells that they can fire. When they run out of shell they try to avoid being hit by entering the evade state unless there is an ammo create on the floor that can be picked up, if they pick up the ammo create then it will reset their ammo.

# Additional Content

Picking up Entity: In my project I allow the user to pick up every entity on the screen apart from the floor. I allow this to happen as it can make for some interesting in counters. If it’s a perfect 1v1 situation and the tanks can’t get a shoot off because of the building then I can just move it aside or if an ammo create if following down and I don’t want red team to have it then I can just move it closer to blue team.

Way point representation: In my project each patrol point is represented by a small quad on the map, green team has green quads and red team has red quads. With the picking up mechanic the player can move the quads while also moving patrol points too

Health Creates: Just like the ammo creates there are health creates that take long to drop but will help the tanks take more hits, while the ammo creates take 20 secs the health creates take 30 secs and will heal the tanks for 50 HP.