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.

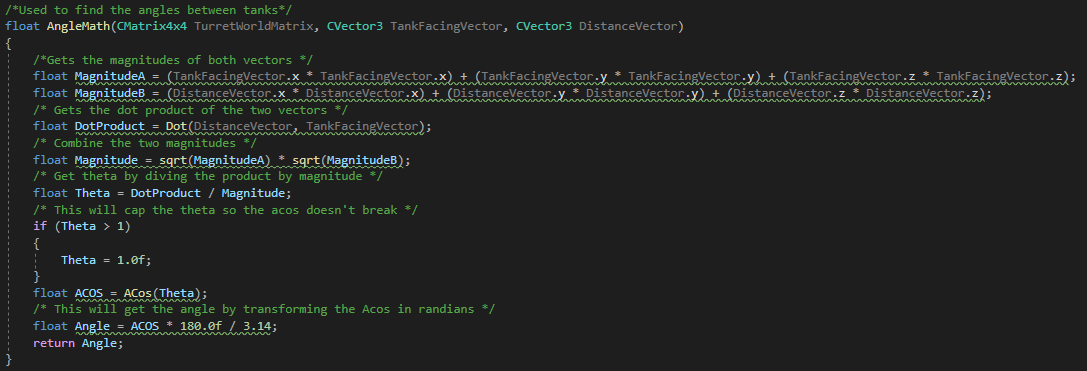
Movement Behavior

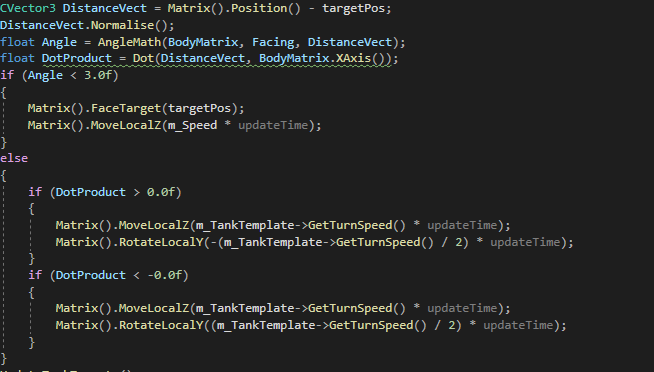
When I comes to movement in the patrol state I use a combination of angle math’s and dot product to determine if the tank need to turn left, right or move forward.

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.

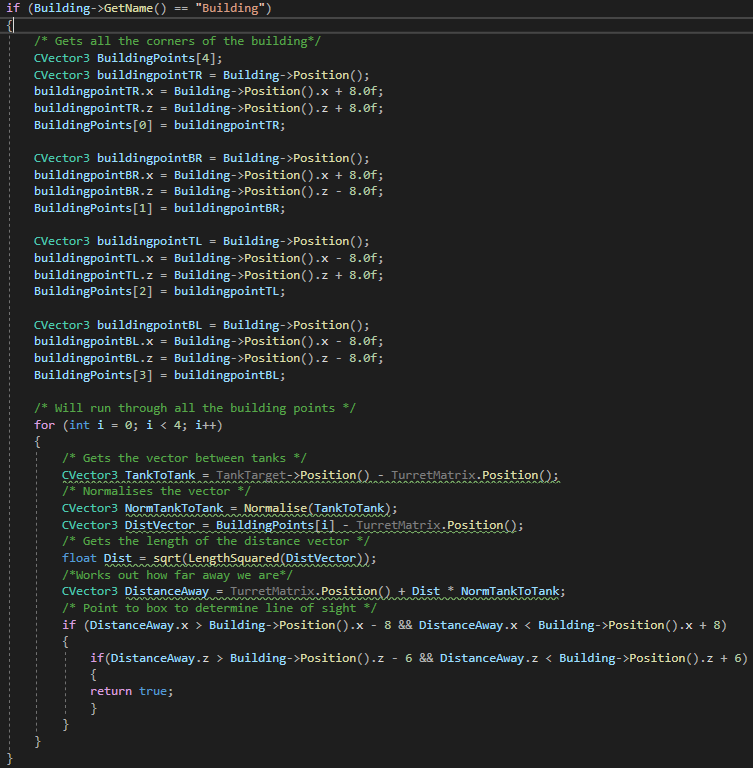
The next step is to Acos theta, but to make sure the Acos doesn’t return a invalid number I cap it so this isn’t possible, after we have the Acos we times it by 180 divided by pi and this will give us the angle.

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.

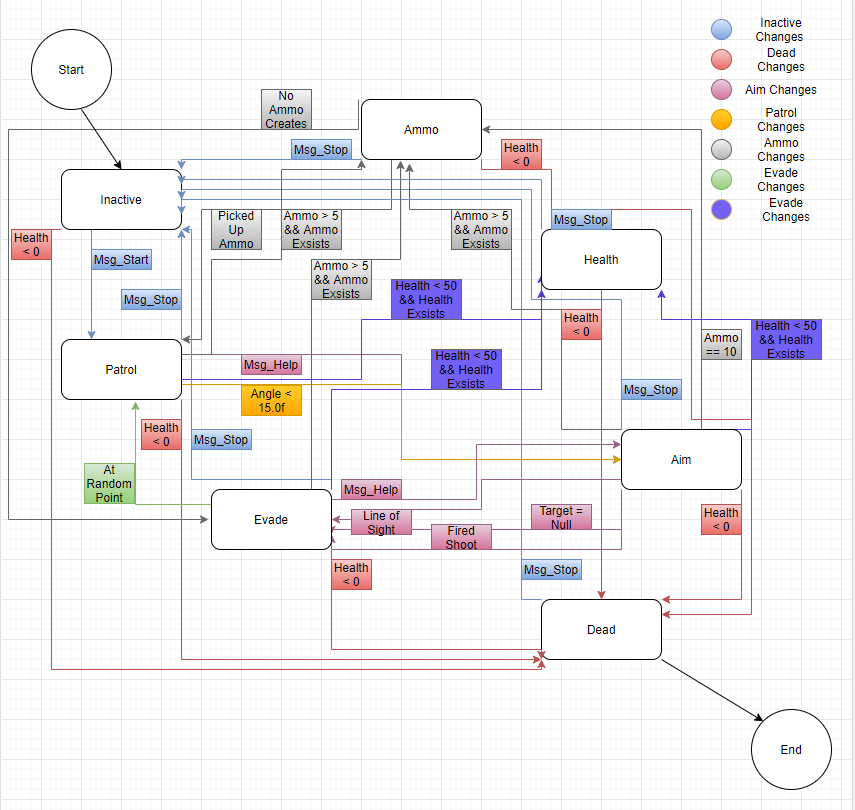
I loop through all the scenery until I get a building, I get all the corners of that building. Once I have all four points then I run a for loop on each point.

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:

Use of Messages:

# Implementation of Update

# Additional Content