Tank Game Behaviour Tree Demo

For my Final coursework, I decided to take the Tank Game demo from lab 1 and expand on it. My initial demo for lab one had extremely jagged movement, fire when the player would be behind the player should the conditions were met and even fire when a building was directly in front of it – directly killing itself. I wanted to expand on this demo to fix all these problems as I would like to use NPBehave for my final project to create many NPCs that behave differently.

I wanted to make the AI to 'patrol' the area before chasing the player tank and shooting it. To do this I created a BlackboardCondition called 'targetOpen'. This allowed the AI to check if the tank was out in the open and not behind any of the environment such as a building or a wall and follow it if this was the case and aimlessly move until it so. I did this by using sphere casting which is a technique used to detect objects at a distance. The sphere cast sends out a 'signal' in the shape of a sphere to the location of the player tank and if the player tank was out in the open, the player tank will get 'hit' by said sphere and return true (if the game object contains the name "tank") thus prompting the AI to chase the player. However, if the player was not out in the open and behind a building the AI will simply move forward and navigate the environment.

In my original demo, the tank AI would move in a zig zag pattern due to the constant change in direction which was a problem since I wanted the movement to look smooth. This was tricky at first however with time I managed to fix it by properly implementing the composite nodes 'Selector' and 'Sequence'. This allowed me to place BlackboardConditions within BlackboardConditions which in turn allowed me to place specific instructions on the AI such as moving forward and turning – avoiding the buildings and walls *only* when the target was *not* out in the open but then following/shooting at the player when it was in sight. By using the composite node 'Selector' I was able to give the AI smoother movement (when patrolling) by making it only turn when specific conditions were met as shown in Fig 1. As you can see in Fig 1 if there is no part of the environment on the left of the AI it moves backwards and turns; this is to prevent the AI from getting stuck, forever moving towards a part of the environment. To even detect the environment, I opted to use ray casting as opposed to sphere casting because I did not need the cast to fit through gaps to see the player. The ray cast emits a line from the AI at a certain distance which then collides with the environment, returning true.

Using the same technique, I created the movement for the AI so that it chases the player. If the player was out in the open and was hit by the sphere cast, the tank AI would directly chase the player until it reached a certain distance (35 – which I found was roughly the longest range of the maximum force for fire by trial and error - I found this by moving to the position when the shell was fired at max force.) The tank AI checks to see if the player is within 35 meters however will only start moving once it is it directly facing the player tank which it checks by checking if the player is *not* in front of the enemy AI. If the player tank is not in front of the enemy AI then it will stop moving and turn towards the player in a similar manner to it turning according to the environment; once the player in directly in front of the enemy AI, it stops turning and proceeds to move forward until in range. Once in range, the enemy AI will shoot the player tank using a force that will hit the player directly, I found this force by trial and error in the same way that I found the max distance the shell travelled. Once I had the distance travelled I simply wrote an equation to find what the distance between the two tanks needed to be divided by to get a force between 0 and 1:

$$x = distance/force$$

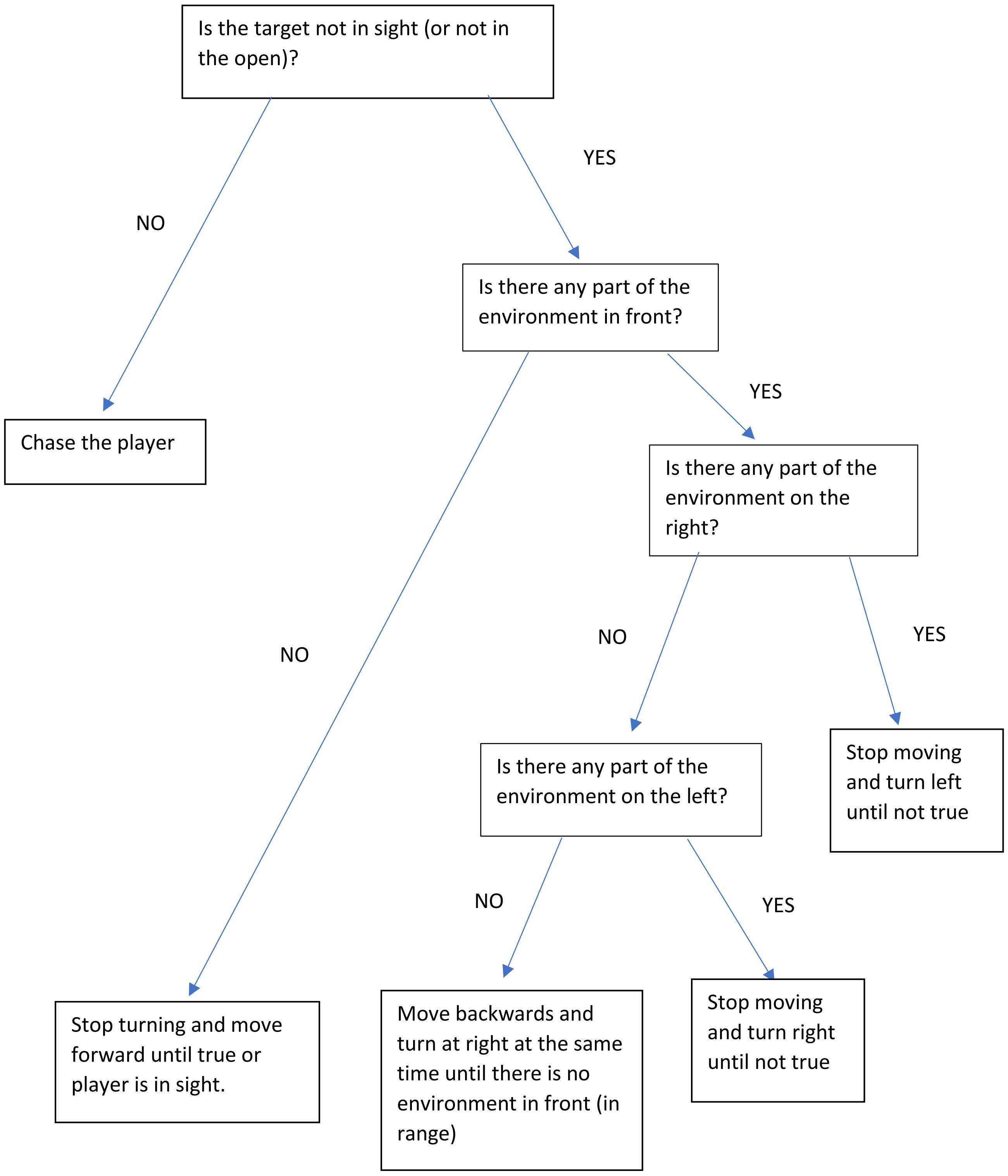
I tried several force values and tested each one to find a distance I came to the value 60. From there I simply had to create a float function called force that calculated the distance between the two tanks and divide it by x = 60 which is then plugged into a new node Fire.

To make the game even harder, I made the AI stop once it was in range and only turn towards the player while shooting. This doesn't give time for the player to stop and gauge their shot.

For my path finding, I implemented ray casting which I learnt through watching the following Youtube tutorial: https://www.youtube.com/watch?v=XYIyRKI3HW0&t=1924s

Naturally the unity reference helped a lot!

Fig. 1 – patrol movement if the player is not out in the open.



Is the target not in sight (or not in the open)?
NO
YES
Chase the player
Is there any part of the environment in front?
NO
YES
Is there any part of the environment on the right?
NO
YES
Is there any part of the environment on the left?
Stop moving and turn left until not true
NO
YES
Stop turning and move forward until true or player is in sight.
Move backwards and turn at right at the same time until there is no environment in front (in range)
Stop moving and turn right until not true