AI BEHAVIOUR EVALUATION

SP2 Assignment Reflection

ABSTRACT
Evaluates my AI assignment and covers the ease, performance, and improvements of the assignment

Name Withheld
Artificial Intelligence

Introduction

This document will detail the various aspects of the Artificial Intelligence AI Assignment. It will discuss implementation of the Assignment, what went well what did not go so well. I will also the discuss the performance of the current assignment, what is working well, what isn’t, what is yet to be implemented. Finally, I will give a personal Post Mortem on how I believe I went on the assignment.

Implementation of AI Elements

Agent

The Agent implementation was first and it was also the easiest as this was already created from previous works, so there was plenty of reference when importing the class into the assignment.

Player

The player class was easy to implement as again I had plenty of reference from previous tutorials. The player had a simple keyboard controller behaviour.

Enemy

Enemy was a constant battle of changes, and those changes were causing the enemy class to break. Enemy was the code that was changed the most as this assignment is focused on AI and the enemy is completely code driven. Initially everything was implemented fine, but as I got deeper into the assignment there was a lot of re-writing code and moving the behaviour tree around. I will discuss the Behaviour Tree separately, but, this agent probably could have used a Finite State machine to achieve what I wanted.

Behaviour Tree

The behaviour tree was a constant battle, especially getting deeper into the assignment. Maybe it was my lack of understanding of what a behaviour tree is and how it was to be implemented properly, but the tree never really work as I would have liked it to.

First, I checked my designed AI tree and quickly realised that only the Patrol Branch of this was going to work with my project setup, so the tree naturally evolved to Figure 2, which is on the next page.

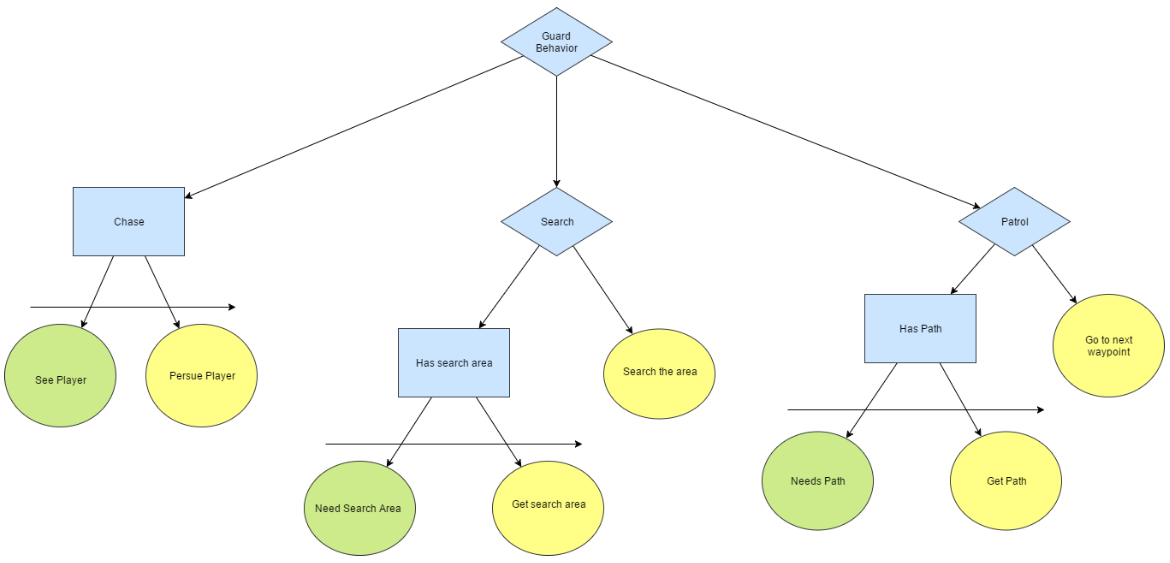

Figure 1: Initial Behaviour Tree

So, this how the Enemy behaviour tree ended up. There are obvious changes, but notably, the first tree forgot any attack behaviour of any kind, and when I implemented the above tree, I realised that chase and attack would have to be sequence child's off a different selector composite. Hence seek composite was created.

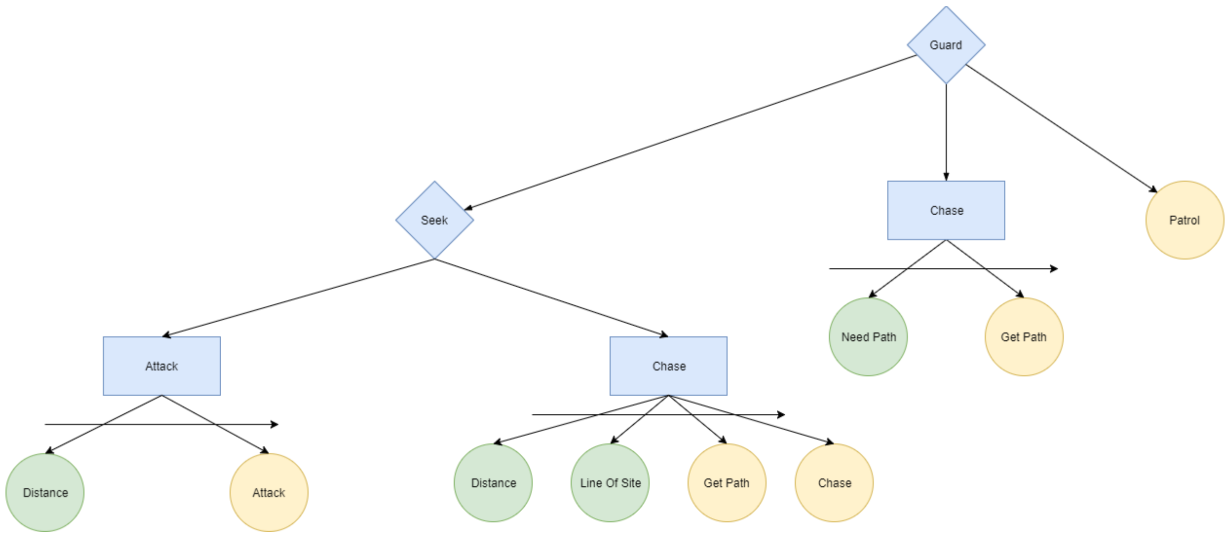

Figure 2: Current Behaviour Tree

The above tree works for the most part, however it has the constant issue of the AI Agent occasionally, not being able to return to patrolling after chasing the player. Despite my efforts I could not rectify this.

Pathfinding

The pathfinding algorithm itself was implemented easily as I had references from previous projects. It is simply implemented in the Graph class and it will find a vector of points. Only A* was implemented and three heuristics were implemented.

- Distance
- Manhattan Distance
- Djikstras (no Heuristic)

I should have looked at implementing different heuristics for this, as I feel I could have optimized the game more by choosing better heuristics for the various times pathfinding was used.

Once the path is calculated the agent will then follow the path and when it gets close to the current point in the path, the agent chooses the next point. This was difficult to get right, as if the distance was too long, the agent would run into walls and if the distance check was too short, the agent would not move onto the next point in the path.

It was easy enough to make the AI Agent find a path to waypoints. The heuristic used for this is Manhattan distance, this being a 2D game and the graph nodes are not diagonally connected. I think if I was to do the assignment again, I would plot a lot more waypoints and only use pathfinding to navigate the rocks and other obstacles. Right now, the agent has about 2 to 4 waypoints, and the pathfinding handles the rest.

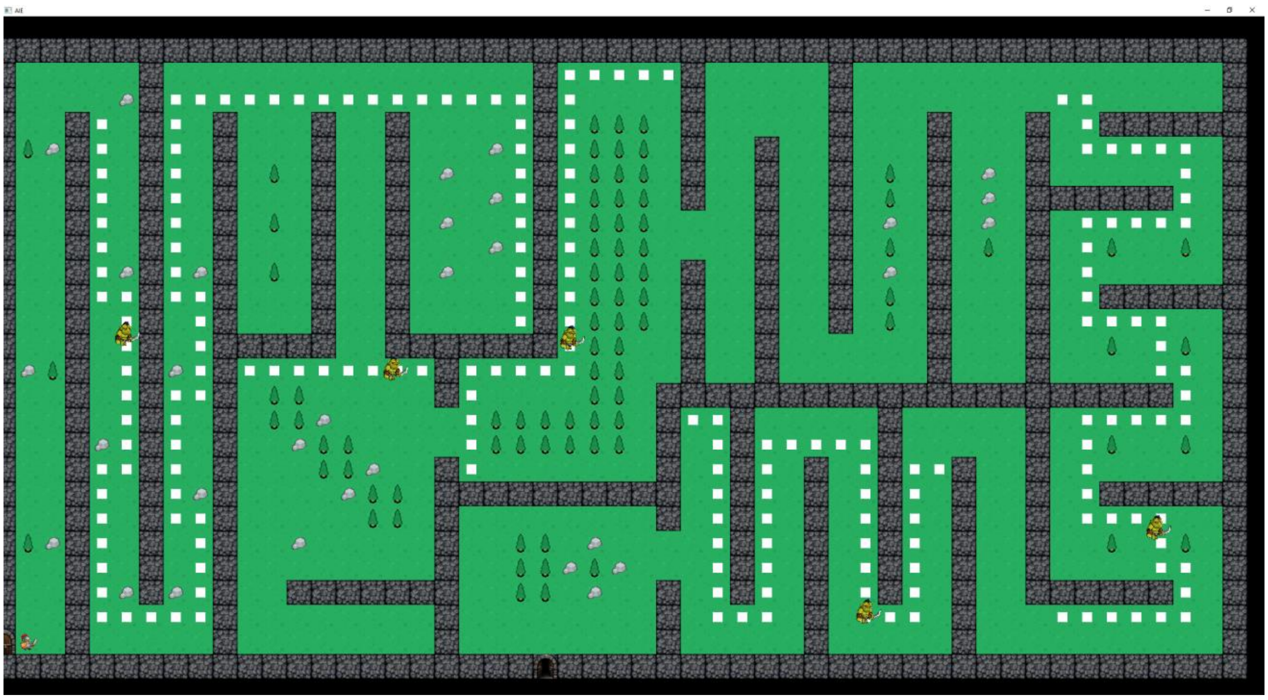

Figure 3: Waypoint Pathfinding

Pathfinding to the player was a little more difficult and in turn has caused Debug mode to crash occasionally, as the calculations slow the game down and cause the player to jump around quite a lot. On Release mode, it works fine. The heuristic used in this case was distance. The main issue with implementing this was a bug where the agent would run into walls rather than path to the player. This was resolved by simply switching the start and end node in the A* Algorithm.

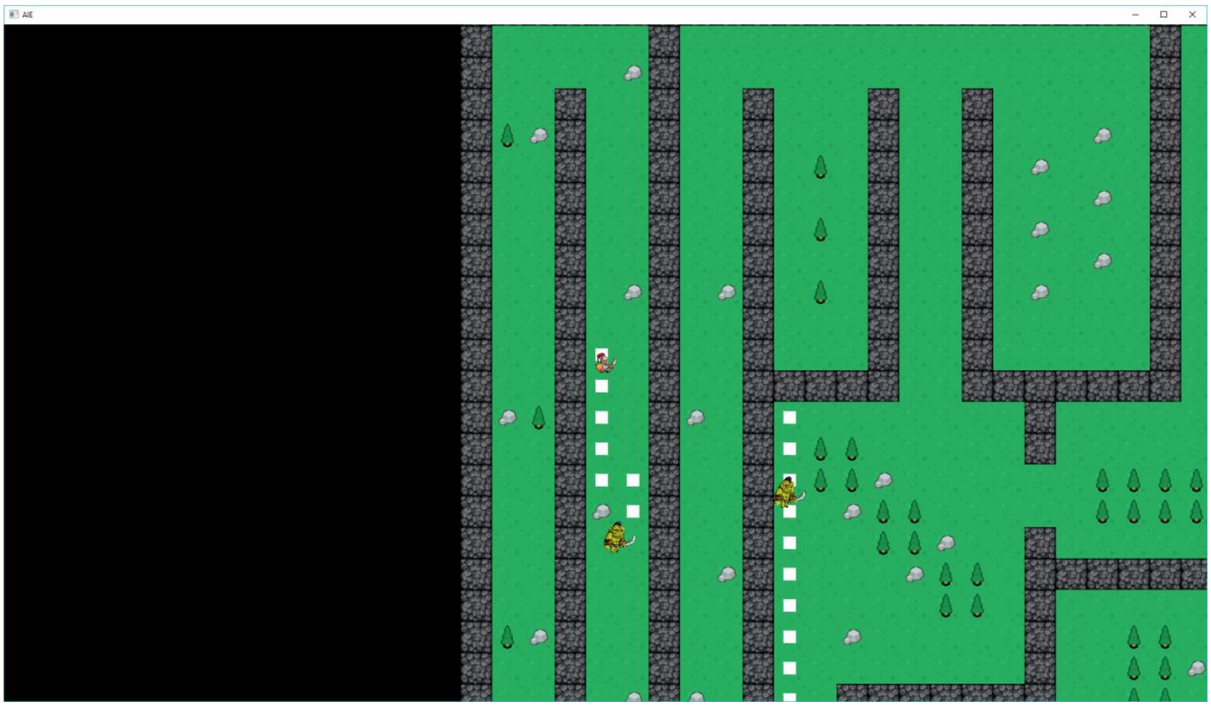

Figure 4: Player Pathfinding

Collision Detection

Collision detection initially was hard to get going as I was not entirely sure how to implement this correctly. Initially I was going to create an AABB class and attach that to nodes and agents. In the end, I decided to implement various collision checks in the game class itself. It takes an agent and determines if they are colliding with a node that is impassable. If the node is impassable they cannot pass.

I made this more difficult on myself as I didn't create a base game object class for ALL objects. This would have made the entire process so much easier as I could have set size variables and would not have had to have multiple checks in my update function, saving processing power.

The following resource was used heavily when creating the collision checks
https://learnopengl.com/#!In-Practice/2D-Game/Collisions/Collision-resolution

Line of Site

This would prove to be a very difficult mechanic to implement. I started with Bresenhams line drawing algorithm, and got this working quickly, however I felt a pixel perfect line for this game was overkill, and I could not determine an effective way to see if the line had intersected an impassable node.

I decided to implement raytracing on a grid, which seems to be adopted from Bresenhams anyway. My main resource was http://playtechs.blogspot.com.au/2007/03/raytracing-on-grid.html. This allowed me to then create a line basically using node positions, thus I was easily able to compare the current position of the line with a node in the graph and determine if that node was impassable.

Performance

What is working?

The keyboard behaviour is working as expected.

Pathfinding to waypoints, is also working well. The enemy agents seem to find their way back and forth without too many issues at all. They are also able to path around objects quite well.

Line of site is also working well. The agent can see the player when in view and will not see the player if behind a wall.

Collision detection is also working well. The enemy agents and the player agent are not able to pass through rocks and walls. The player gets slowed when going through trees. And the player can win the game by colliding with the open door.

What is not working?

The behaviour tree itself is affecting the AI agent’s ability to go back and forth between chasing and patrolling. The path back to the WP is not working as expected, and all efforts to fix this did not work. It must be said it only happens occasionally, but a bug nonetheless.

While collision detection is working, it would have been nice for the enemy agent to automatically move around the object or choose a different path if it got stuck.

Debug mode is also having plenty of issues. This is cause the amount of calculations that are occurring. This is causing the game to jump as well as the player positioning, sometimes breaking the game.

Improvements

Search

This really disappointed me that I did not manage to implement this. The agent was meant to pick 2 or three random nodes around the node the player was last seen on and patrol them. Once finished the agent would return to patrolling its allotted waypoints. Really unhappy this is not implemented, and a definite improvement to be added!

Field of vision

Right now, to determine if the agent is visible the Enemy uses a combination of distance and Line of site to determine if the player is in view. I would change this so that it would only chase players in front of it. The preference for this would be a cone in front of the enemy agent, and if the player is in the cone, do a Line of site check then chase.

Shadow Mapping

While I think this is probably not necessary and a bit over kill, it would be cool if the player could only see nodes that were in the range. Using shadow mapping would have made a really cool effect.

Heuristics

I would invest more time, checking for different heuristics that could be used in A* to optimize the pathfinding of the Agent. Currently only have three basic ones, I feel there are better options out there.

Improved Behaviour tree nodes

Would look to improve the behaviour tree nodes so they include a pending result. This would probably make the patrolling and chasing work so much better, as I could have set the node to pending until the path was completed.

Game Object class

Having a base class for all objects that had a position, size and texture would have made some of this much simpler. Will remember to implement this next project.

Post Mortem & Conclusion

Overall, I am fairly disappointed in the result of this assignment. It never really worked anywhere near where I wanted it to. But it does ‘work’. There is just so much left out, and so many bugs unsquashed. I am happy that I could achieve pathfinding as well as challenging myself to make a behaviour tree. But at the same time neither feel like they are implemented as well as they could be.

Personally, I need to do better in future assignments. Too often I told myself that “I only need to implement ‘x’ things”. Only to not consider that ‘x’ things could have a list of things that range from a – z. This has caught me out so many times in programming, and was really evident in AI. It caused me to fall behind and not be able to seek the necessary help from Teachers. In future, I need to plan things better, as well as write notes while developing, as there is so much to consider it is easy to visually see what needs to happen.

Maybe I am being too harsh on myself as I am still new to programming, but I feel if I am critical I will learn from my mistakes and improve in future projects.