# The Simulated Environment

This project simulates a simple battle between two non-playable avatars.

The simulated environment includes two avatars with separate similar decision trees attached for decision making. As seen in the diagram shown below (A), the avatars each have 3 states: Wander, Chase, and Attack. These states transition between each other by decisions such as:   
Whether the enemy is within the chase range, or whether the enemy is within the attack range.

Also included in the simulated environment is a node map which allows the avatars to navigate around the environment, it is visualized very simply with the navigable areas visualized as a grid, and the unnavigable areas filled in with grey walls.

The avatars use A\* pathfinding to generate a path along the node map which is then smoothed. Currently the node map is hardcoded to have a few rooms and a block for the avatars to circle as seen in the screenshots (C) and (D).

The main parameters of the simulated environment are the avatar’s speed, health, attack damage, and attack time which is just how long the avatar will take to attack. The conditions also take in various parameters such as the enemy distance condition taking in the distance for how close or far the avatar should be to transition to chase or attack.

As seen in screenshot (B), upon starting the simulation the user is prompted to select a difficulty which is represented as a number between one and four. This number is then used to control various parameters such as each avatar’s speed, health, attack damage, and attack time, along with condition parameters such as how close or far from the enemy the avatar should be to transition to the chasing state or attack state.

## The A.I. Strategy

|  |
| --- |
|  |
| (A) A diagram of the decision tree. |

The diagram above describes how each state transitions from one and another and by which decision it will make to do so. Each state is visualized by the avatar switching colours such as green if wandering, or red if attacking, this is also represented above by the diagram.

The wander state works by picking a random point on the node map and navigating towards it using pathfinding.

The chasing state is similar, though instead of picking a random point on the node map it will instead use the enemy avatar’s position and generate a path towards that instead so the avatar can navigate towards the enemy.

The attack state is the most complex state. When the avatar enters the attack state, the avatar’s attack damage is randomly chosen along with the attack time, both are controlled by the simulation’s difficulty. Then on each frame, it’ll increment a clock checking if it reaches the attack time, when it does the enemy will take damage and a flag will also be set so that the enemy doesn’t take damage each frame the avatar is attacking, the clock is then reset.   
The attack state has a chance of picking 0 as the attack damage allowing the avatar to miss. After dealing damage to the enemy, the avatar will then flee from the enemy, as to avoid getting stuck within the attack range, the fleeing works the same as the wander state.

The conditions are hopefully self-explanatory, such as the enemy distance condition which just checks the distance between the enemy and avatar based on the parameters.

# Screenshots

|  |
| --- |
|  |
| **(B)** Screenshot of the difficulty selection prompt. |

|  |  |
| --- | --- |
|  |  |
| **(C)** Screenshot of the two Avatars fighting. | **(D)** Screenshot of Avatar 1 winning the fight. |

# Feedback

|  |  |  |
| --- | --- | --- |
| **Feedback** | **Problem** | **Changes Made** |
| Add filters to organise files in the solution explorer within the Visual Studio project. | There were no filters created to organise the source files and headers inside the solution explorer, so it was a mess to find anything. | Added filters to separate decisions, conditions, pathfinding, and game files to those categories. |
|  |  |  |