# The Simulated Environment

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

The simulated environment includes two avatars with separate but similar state machines attached for decision making. As seen in the state diagram (A), the avatars each have 5 states: Idle, Wander, Chase, Flee, and Attack. These transition between each other using various conditions such as: At random, whether the avatar’s health is low, whether the enemy avatar is in range or not, or whether the avatar is currently attacking or not.

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, the random timer condition taking in the minimum and maximum values to pick between, and the low heath condition taking in how low the avatar’s health should be for it to be able to transition.

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 the various condition parameters such as how close or far from the enemy the avatar should be to transition, and the max value of the random timer condition.

## The A.I. Strategy

|  |
| --- |
|  |
| (A) A diagram of the State Machine. |

The state diagram above describes how each state transitions from one and another and by which condition it will transition. Each state is visualized by the avatar switching colours such as pink if idling, or red if attacking, this is also represented above by the state diagram.

The wander state works by picking a random point on the node map and navigating towards it using pathfinding, the fleeing state works the exact same way with the only difference being that the avatar’s speed is doubled while it is in that state.

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.

The idle state simply does nothing at all and is just there to pause the avatar.

Most of 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, the random timer just picks a random value between the minimum and maximum parameters and counts down to it using a clock similar to the attack state, transitioning once it does.

# 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 |
| Repetitive & confusing FSM design. | In my original design, the finite state machine for each avatar in the environment would transition back and forth between wandering and chasing on a set timer, only transitioning to the attack state when they are close together before transitioning back to wandering again with on a set timer. | Redesigned the FSM design, adding new behaviours and conditions such as an idle behaviour, flee behaviour, and a low health condition, random timer condition, and a condition to check whether the avatar has finished attacking. |
|  |  |  |