ZERO TEAM - Assignment 4: AI

Team Members: Felipe Paz, Seth Parsons, Thomas Moore

AI System Design:

For this assignment, we plan on having two types of AI: a friendly AI and a hostile AI. The friendly AI will wander around the map and try to collect tokens for the player. After collecting tokens, the friendly AI will find the player and return the tokens to them. The hostile AI will wander around a fixed point. If the hostile AI spots the player, it will move towards the player and attack them. After attacking the player, the hostile AI will retreat back to its starting location to "recharge" after an attack.

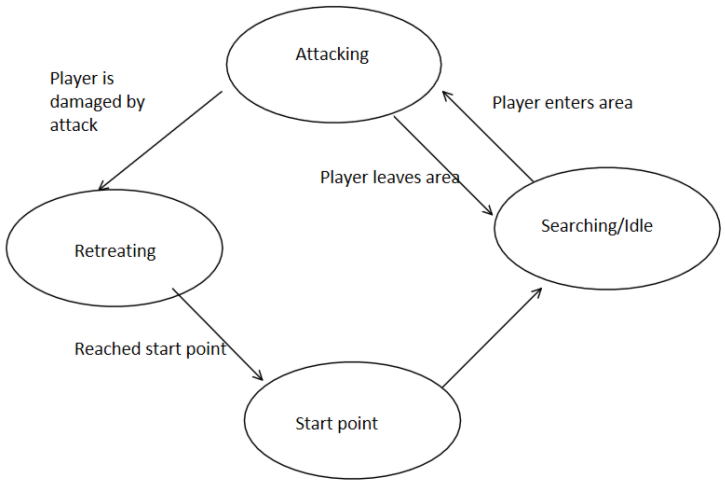
We plan on implementing our AI using a finite state machine. Each AI will have three generic states: an idle/searching state, an action state, and a fleeing state.

For the hostile AI, the three states will be searching for a player, attacking the player, and retreating to its starting point. Below is a diagram showcasing which actions trigger the finite states and the relationship between each state.

Enemy AI:

States:
Searching/Idle
Attacking
Retreating

Events:
Player enters area
Player is damaged N times by attack
Player leaves area
Reached start point



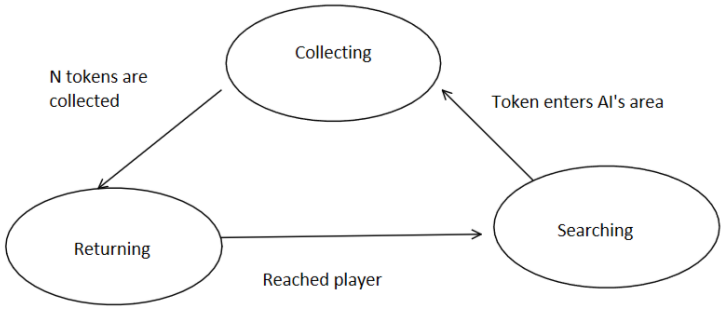
One design consideration we are currently thinking about is how long the hostile AI will chase the player for in its attacking state. Since our player has no way to attack back, we may have the hostile AI move slower than the player, so that our player has a way to avoid attacks.

For the friendly AI, the three states will be searching for tokens, collecting tokens, and returning the tokens to the player. Below is a similar diagram as the previous for friendly AI behavior.

Ally AI

States:
Searching for tokens
Collecting tokens
Returning to player

Events:
Token enters AI's area of "visibility"
Token is collected by AI



In terms of implementation specifics, the first thing we are going to do is abstract out our player's movement actions into their own methods. Currently we are handling most of our movement all inside of our process and physics_process methods. We will break actions up into a move(), a jump(), and other movement methods such that our AI scripts can call these instead of just rewriting that functionality. We're not sure exactly how we will implement the logic for triggering different types of movement (jumping, gliding, ledging) for our AI currently.

After that, we are going to design our AI models. We currently have two ideas: making areas part of the AI models that trigger functionality once the player enters them, or having areas part of the level that the AI responds to once players enter them.

With regards to pathfinding, we will likely be using Godot's built in AStar object. We will have a point for the models of our AI and for the player model, and connect these points together into segments for the pathfinding. For our friendly AI, we will have an additional point to tokens in the level. These points will be added upon contact with the AI's areas.