Final Project Report

The problem we decided to tackle was the creation of an AI that you could battle in a game. The AI will be able to select moves depending on the state that it is in. The states that it can be in are aggressive, defensive, and balanced, and each of these states include a certain percentage chance to attack, intimidate, heal, or defend. The AI should not take any actions that have no effect. For example, it will not heal if it is currently at full health. This problem is important because it is an interesting introduction to the creation and implementation of AI. The first steps to creating a fully functioning AI in the real world is by first creating one in a virtual one. Our motivation for pursuing this problem is that we are both interested in the topic of videogames so we decided it would an engaging project/problem. The AI will use a state-space search to decide which state it should be in when deciding to make a move. After the AI chooses a state it will have a weighted chance at choosing one of the 4 available moves depending on the state it chose. For example, an aggressive state has a higher chance of attacking versus the defensive state which has a higher chance of defending. The specific input that we are putting into the problem is the move we choose against the AI, and the output will be the stance that the AI chooses and the move that is chosen. The application of AI that our project falls under is the game playing application which is considered a part of the formal task branch of applications of AI.

The input we used to carry the process of the AI is all done within the command prompt, and it is the moves we make during the game. The AI observes what the move we make does and switches to a stance depending on what that move does, and what moves it could make in the future. After it switches to a stance it has a probability of doing one of four moves, and the probability of each of those moves changes depending on the state that the AI is in. For example, if the AI chooses to be in the aggressive state, it has a 60% chance to attack, a 20% chance to intimidate, a 10% chance to heal, and a 10% chance to defend. The reason we used probability in the stances is because of the nature of the project. A game is more interesting if there is some form of chance because the player will never know exactly what the AI will do. If we didn’t use chance it would be fairly simple for the player to guess the moves of the AI due to the limited amount of options. These are the specifics for each state:

|  |  |
| --- | --- |
| Move | Percentage Chance |
| Attack | 60 |
| Intimidate | 20 |
| Heal | 10 |
| Defend | 10 |

Aggressive

Defensive

|  |  |
| --- | --- |
| Move | Percentage Chance |
| Attack | 20 |
| Intimidate | 10 |
| Heal | 35 |
| Defend | 35 |

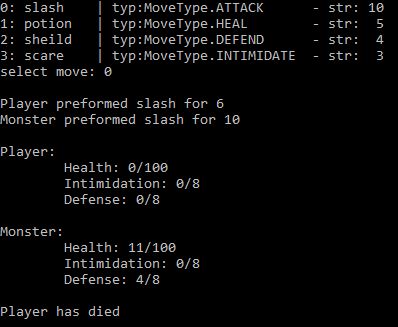
Balanced

|  |  |
| --- | --- |
| Move | Percentage Chance |
| Attack | 30 |
| Intimidate | 20 |
| Heal | 30 |
| Defend | 20 |

The AI chooses which state it will use to make its move by comparing the health values of both entities (itself and the player) and it also simulates the next round and makes a decision based on that simulation. For example, if the AI simulates the round and finds a path in which it kills the player, it will take that path. Otherwise it will search for a path that provides the best outcome for the later rounds. In the preprocessing stage of the AI we specifically stated that the AI could not do any move that technically does nothing. An example of where this applies is if the AI chooses the balanced stance and randomly chooses to heal when it is at full health, the AI is forced to do a different action.

The method we used in this project is state space searching and simulating future events to find the path that gives the AI the best chance at winning. The idea behind a state space search is that the AI is looking for our specific goal state, which is to kill the player, and chooses the best path to reach this goal. How this specifically works in our project is that the AI chooses a fighting stance based on the current conditions of the fight using both entities hp values. It then simulates the next few rounds and calculates the final health results of these simulations and makes it’s moves based on these simulations. It runs another set of simulations after each move the human player makes so it adjusts its choice of move after every inputted move from the human player, and if it ever has the opportunity to kill the player it will choose that move no matter the circumstances. When running a simulation and arriving at a killing move it will choose that move no matter what because it will allow the AI to reach its goal state of killing the player.

The experiment we used to test the AI and how it responded to our moves was using specific strategies when battling the AI. For example, if you repeatedly use only attack moves when battling the AI, it uses healing and defense moves to try and reduce the oncoming damage and to regenerate its health values. If you use the strategy of trying to up your defense, the AI reacts by repeatedly attacking because it is not suffering damage and doesn’t require healing or defense. When doing testing the minimum number of moves before one entity was defeated was 15, and the maximum number of moves was approximately 50. The average number of moves per game was around 30 moves for one entity to win. The main outtake from our experiments was that even when repeating the same strategy over and over again, the result of the battle is not always the same due to the intentional calculated randomness when selecting moves. Experimenting with this algorithm has an extremely long output due to the amount of moves each entity makes, but here is an example of the final move the AI made when I used the strategy of only attacking:



This picture shows the full interface of the program, and what happens when the AI wins. It shows the selection process for the human player who can choose between 4 moves. The next section of the picture shows what move each entity chose and what it did (in this example both entities chose attack). Finally, it shows the final health, intimidation, and defense values of each entity. In this specific scenario, the player died because his health value reached zero before he could kill the monster. This is just one of many tests/experiments on the AI. The other experiment we did was messing with and altering the values of health, intimidation, defense, and each of the four moves until it felt like it was right. By playing around with any of these values it could dramatically alter the course of the game. For example, too much attack would not give the AI an opportunity to respond due to very short games, but too little attack would make the game extremely long, and potentially infinite due to the healing function. Many of these values were changed many times throughout the process until we felt like the game was somewhat balanced, but with even more time we could balance out the game more.

In conclusion, our AI is a fully functioning opponent in a turn-based fighting game. It can decide moves based on the AI’s stats compared to its human opponent. It can also simulate future rounds to try and figure out the best path for it to take to win the game. Due to the implementation of a chance for it to make other moves, the game is not impossible while still requiring strategy to win. The state-search function used to find the best path to victory is not perfect, but it is the highest performing algorithm used and it was the best fit for the problem we are addressing. If we did not add the random chance of using moves other than the best move to win, it would have created the problem of the game being too difficult or impossible to win. In the future, we could expand the state system to more than 1 on 1 battles, and we could create customizable move-sets to make each execution even more unique. Another thing we could do in the future with more time is create varying monster types with different tendencies because we have currently only implemented one. Another thing we could explore with a lot more time is creating a way in which the AI observes the player’s previous moves to try and adapt to the specific player’s playstyle. The final area in which we could improve the experience of this project is creating varying difficulty levels in the game. We could do this by limiting specific information that the AI would receive. With a lot of time we could create something very interesting and complex extending from our original concept.