Kenneth Willeford

CSCI 164

Adversarial Search in Game Environments

Personal Background

I grew up in a rather poor family with my mother, a father, my brother, and my sister. We were well below poverty level for a long while but eventually we were stable enough to move away from ramen and into meeting non-basic needs.Our family saved up enough for a SNES which would be the first game system my family owned. But even with this new form of entertainment games were still expensive and so on the rare occasion we could get a game we would have to determine if it were worth it. Game reviews and word of mouth were great ways of figuring out the quality of a game. Because replayability was an issue we tended towards RPGs and Adventure games. (Some of my fondest memories were playing The Legend of Zelda: A Link to the Past.) Eventually one of my mom's boyfriends had a computer with windows xp. My brother and sister had moved out and instead of playing games I figured I could have fun making them so I begged and received RPG Maker XP for Windows. RPG Maker utilizes a grid based front end with pseudo programming 'events' to construct a game. However RPG Maker also has access to RGSS (Ruby Game Script System) on the back end which served as my introduction to programming.

A.I. And Games

A.I. in games is often thrown around very loosely by players but we can generalize what type are typically used by genre. In games with dozens of enemies on screen we are likely to see simple reflex agents. Hack and Slashes and Shooters often have large amounts enemies at once

so utilizing anything beyond a simple reflex agent in these situations would be infeasible. However some genres have very few agents such as Fighters or time isn't an issue such as in Turn-Based games however even in these types of games computational compromises must be made. For example in a pixel perfect fighter it may be best to generalize the environment as a grid space so a single look ahead can move much further. Additionally the Agent could commit to the action/action sequence that they have determined for a given period of time to prevent repeated computations.

RPG Maker: Dueling AIs

For my project I will be using RPG Maker to make a dueling AI. The dueling AI can be fought by a player or by another dueling AI. The goal is to explore Expectimax and Minimax on a deeper level. Some important differences between this and the pacman project is that agents require more than one hit to die. As a result some limited learning may be possible. Additionally it would be interesting to play around with generating 'Behaviors' through the searches. For example an Expectimax Agent taking into account their learning would be a more strategic thinker while switching to a Minimax would result in a cocky thinker who generalizes their actions. It would be interesting to abstract these behavioral patterns as the project goes along.

Performance Measure

For Performance Measure it's best to split apart the rational aspect of winning and the humanly aspect of behaviors. For example when it comes to winning what matters is maximizing your own health while minimizing your opponents. However depending on the behavior of the Agent it may determine that it's okay to make a bad trade because it will end their opponent that much quicker.

Environment

The RPG Maker Environment is a finite Grid Space with passable and impassable tiles. As an RPG World Agents will have access to various statistics such as HP and ATK. The agents will be able to operate regardless of their 'Arena.' If there are no impassable tiles then Manhattan Distance will be fine, otherwise a Grid Search will be required to find proper distances and path finding. I will be using a plug in created in the RPG Maker Community, "Pearl ABS Liquid". This will allow a coupling between the Environment, Actuators, and Sensors so that they can more easily get relevant information about each other as well as provide the basis for the Actuators.

Actuators

As a battle simulation the Actuators in this Environment will be four directional movement, guarding, projectiles, and melee attacks. The latter three will form a triangle environment so that one may logically beat the other. Another factor is that every Actuator will have a Cooldown. A period in which that action cannot be performed again. As the project goes along there may be multiple options for a given Actuator Category.

Sensors

The environment will be fully visible. Agents know each other's locations and statistics at all times. However behaviors may be able to be implemented by restricting certain sensors or using approximations instead of exact values to give some error so that it may be more 'humanly' in its action.

Adversarial Search Methods

In addition to what was mentioned before a ‘Psychological Factor’ could be introduced based on punished moves versus good moves to adjust strategies on the fly. Adjusting the game-ply depth will be important as well, to begin with I’ll use a depth of 7 but I’ll test for a good value. In addition comparisons between Minimax and Expectimax can be made in many different situations based on the behaviors constructed.

Evaluation Ideas

Rational Evaluation is simple, a good move is good and a better move is even better. The current relationship is Decisive Move > Safe Move > Unsafe Move > Poor Move. In order to keep the orthogonal relationship between the different types of actuators under control I will classify the actuators into two categories. Safety(Movement/Guarding) and Aggressive(Melee/Projectile.)