Playing Centipede!

Motivation:

I chose to build a playing system for Centipede because one of the earliest experiences I had with any form of A.I. was of a computer playing Snake. Because of this, I wanted to also have my first individual project to involve a self-playing game. I also wanted to gain some experience observing and managing a system observed and reacted to the models I created.

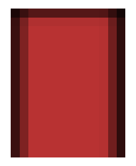
Scope:

The scope of this project seeks to create a reflex-based model to play a round of Centipede as quickly as possible while attempting to maximize it's score gained from shooting enemies. 3 algorithms were made, an optimistic expectimax, a pessimistic expectimax and an a-b pruning expectimax.

Design:

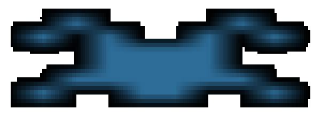
- Centipede is a pretty deterministic game and so my model resembles closely an expectimax model.
- The only random-esque factors are the spider enemies (these spawn and move randomly) and the player movement (movement is a range and a single input may or may not be the same at a different time)
- Average movement distance is about 6 pixels so my expectimax recurs on 3 different distances we expect our ship to move per action.
- The utility of the model depends on the positioning it ends up at and the number of centipedes, mushrooms and spiders it expects to defeat

Ship



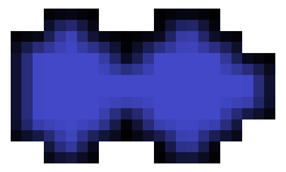
It's **YOU**, don't die!

Spider



It moves randomly and sporadically. Worth **300, 600, or 900 points** depending on distance from ship!

Centipede



It moves deterministically depending on the level it's on, worth **10 or 100 points** if it's the leading centipede!

Mushroom

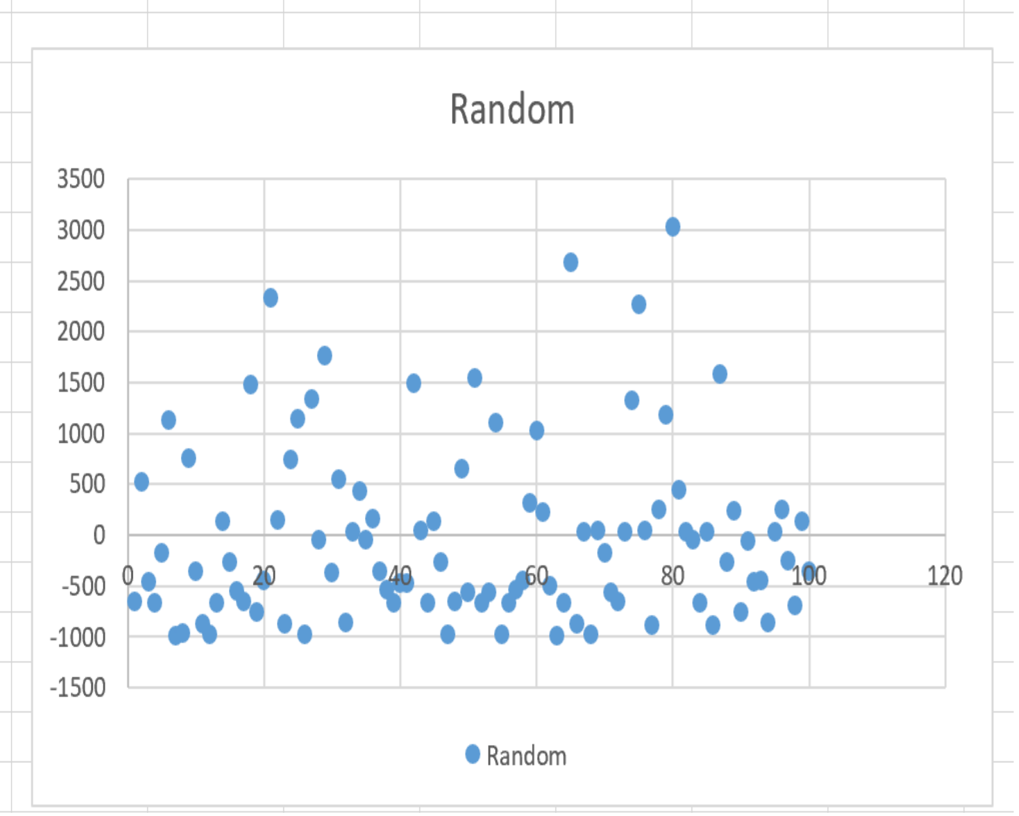
Created at the beginning of the round, additional ones are made after shooting a centipede, worth **1 point**

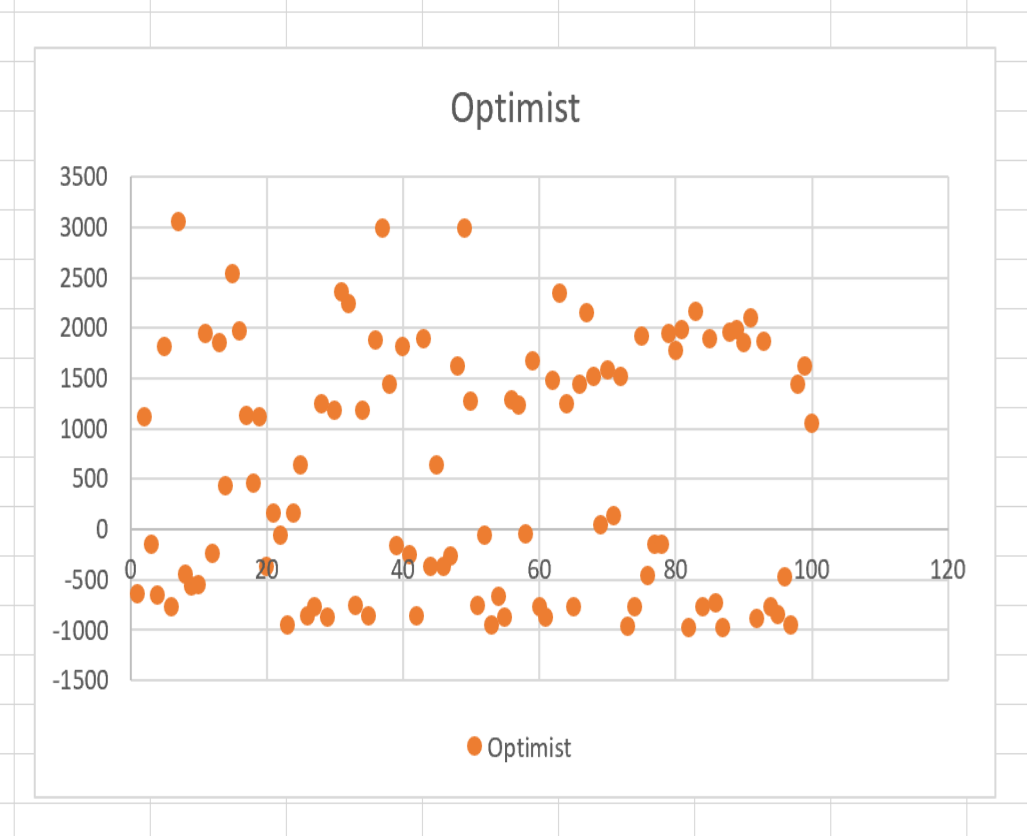
Challenges:

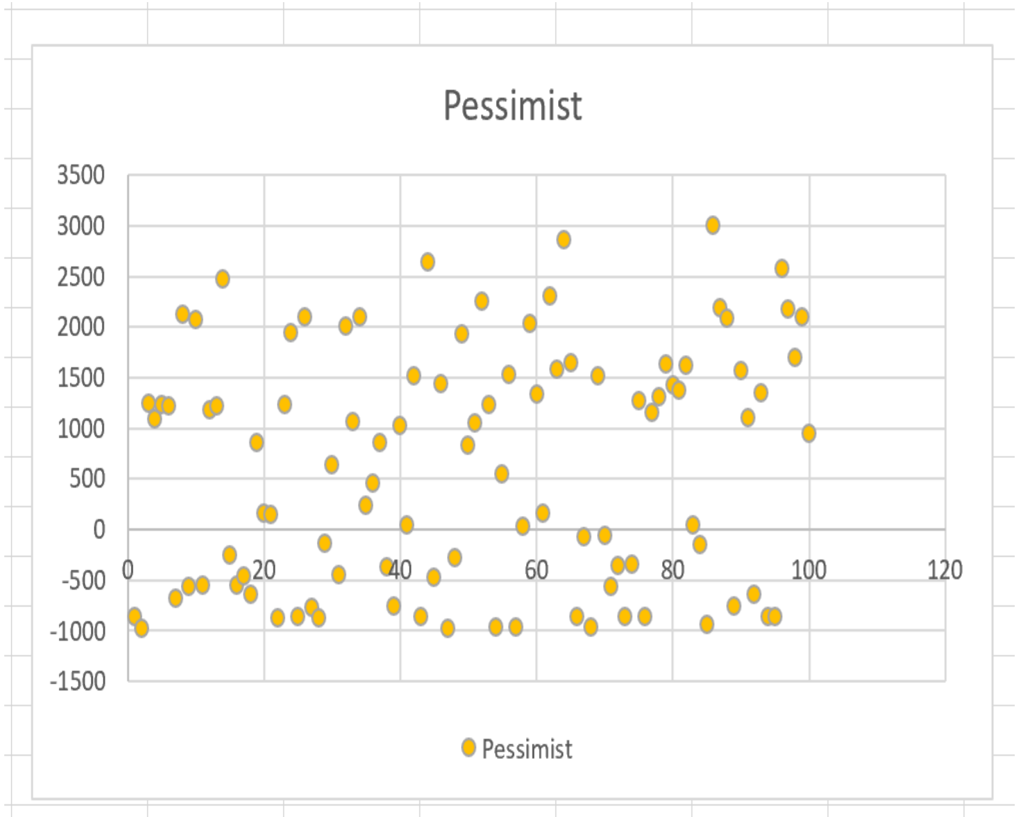
While the model is mostly deterministic, I struggled creating a prediction system which I could use to base the model on. Particularly, reconciling the random movements of both the ship and Spiders in the prediction process took much trial an error. Another struggle I found was in creating a utility to accurately summarize the position the prediction model had ended up in.

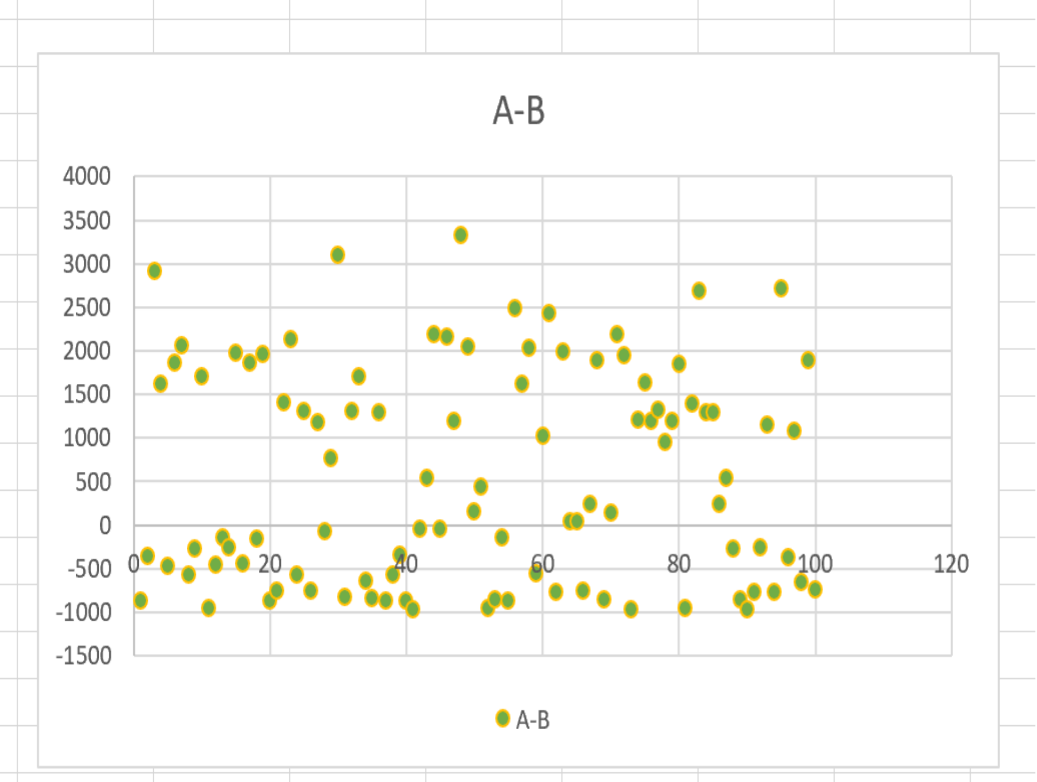
Results (100 Games):

- Random Agent averaged: **-16.63**
  - Definitive Loss (<-900): **8**
  - Loss (<1000): **76**
  - Win (>1000): **12**
  - Definitive Win (>2000): **4**
- Optimistic Agent averaged: **585.86**
  - Definitive Loss (<-900): **6**
  - Loss (<1000): **47**
  - Win (>1000): **37**
  - Definitive Win (>2000): **10**
- Pessimistic Agent Averaged: **632.13**
  - Definitive Loss (<-900): **6**
  - Loss (<1000): **46**
  - Win (>1000): **31**
  - Definitive Win (>2000): **17**
- A-B Agent averaged: **562.68**
  - Definitive Loss (<-900): **6**
  - Loss (<1000): **50**
  - Win (>1000): **30**
  - Definitive Win (>2000): **14**


Random
Random


Optimist
Optimist


Pessimist
Pessimist


A-B
4000
3500
3000
2500
2000
1500
1000
500
0
-500
-1000
-1500
0
20
40
60
80
100
120
A-B