Heart Attack

Caparas, Carandang, Datoc

December 10, 2014

1 Introduction

Our project, entitled "Heart Attack" is a Tower Defense game that features a specialized AI used to generate an optimized sequence of enemies that can be produced given a limited amount of ATP provided for a certain arrangement of towers present on the grid. The main objective of the enemy AI is to eliminate the player, by attacking its 'heart' at the end of the grid, reducing its life points to zero.

2 Implementation

2.1 Game Mechanics

"Heart Attack" is a Tower Defence game with the theme 'Immune System'. The game features the defender, that uses ones' white blood cells and purchases them using ATP, Adenosine Triphosphate. The towers are the white blood cells that defend against the attacker, the pathogens. The pathogens' main goal is to eliminate the player by reducing its life points to zero by attaking its 'heart' at the end of the grid. There are several different types of towers with unique abilities. These towers can be upgraded to stronger towers in order to deal with stronger enemy units. The pathogens, however, increase in lifepoints every two waves by the formula:

$$lifepoints = originallifepoints * fibonacci(\frac{wave}{2} + 1)$$

Each turn, both pathogens and player gain ATP used to purchase their units. The original ATP of the tower and defender is 50 ATP. The defender's gain of ATP is of the form, where x is the number of waves:

$$(0.3x)^2 + 15$$

While the pathogen's gain of ATP each turn is of the form, where x is the number of waves:

$$100log(x) + 100$$

2.2 Single Agent

We implemented several single agent AI to do basic actions such as path finding. Breadth First Search is used for finding the shortest path a wave of enemies will take to reach the goal. Depth First Search is used for checking if placing a tower will block all possible paths to the goal. A Simple Reflex Agent is used to automate tower placing based on the recorded actions of a player, which will be used to train the genetic algorithm.

2.3 Genetic Algorithm

We used a Genetic Algorithm to determine using the fitness function

$$fitness = \frac{(100 - life)}{wave}$$

to create multipliers that will determine the proportion of virus types per wave. These multipliers are fed as weights to an Artificial Neural Network stripped of its hidden layers, technically a perceptron but still uses the sigmoid function.

2.4 Conclusions, Limitations, Recommendations and Future Work

2.4.1 Conclusions

Genetic Algorithm, being based from survival of the fittest, it is well suited for the concept of Agent vs Agent of Adversarial Game Playing.

2.4.2 Limitations

We lacked the time to formulate a better testing method. Our current testing methods only consider some tower configurations which could be improved by providing more tower configurations to accommodate different player styles and avoid overfitting to a specific style. More time would also allow us to train our Genetic Algorithm longer which could yield better fitness. We also failed to consider path recalculation therefore one group of pathogens only takes one path, so in some cases ghost facing occurs if you place a tower in the current calculated path of a group of pathogens. Our path finding algorithm only considers the shortest path, but not necessarily the optimal path, for example, the path with the least towers. We believed that it would make the game too difficult for a lot of players.

2.4.3 Recommendations and Future Work

If ever we decide to polish our game, we would like to impliment more varieties for the pathogens and give more special abilities for them also. We plan to make the HIV attack the towers, while the Ebola will have a special ability to reduce the lifepoints of towers in a certain area of effect around it. We also want to solve the limitation of our AI, where the pathogens phase through towers after the initial calculation. We would also like to improve the AI of our towers to create an optimal build not just for the pathogens but for the defender as well.

3 Criteria

3.1 Difficulty

3.1.1 Coding

The first component of grading for difficulty is coding. For this component, we coded each function, class, mechanic and AI used from scratch. Although we used Pygame as a framework in order to build our project, each class and object were coded all from scratch. The most crucial part of the program, the AI, was not taught in class nor were they any implementations of the genetic algorithm online that were suited specifically for the use of our project. Thus, we can confidently state that our project was coded from top to bottom from scratch.

3.1.2 Knowledge

For the second component, Knowledge, we applied both AI lessons that were discussed in class and used an entirely new AI that was not discussed during the span of the lessons. The AI that we implemented that was discussed during class were, Depth First Search, Breadth First Search, and a Simple Reflex Agent. While the AI that we implemented that was not discussed during class hours is the Genetic Algorithm.

3.1.3 Data Set

For the third component, Data Set, we created our own data set by creating a sequence of actions that would automatically buy and upgrade towers as long as the resources needed to allow that action is available. The data set was created by ourselves as actions when we played and tested our project. This data set was used to train our AI in order to find an optimal sequence of enemies that can eliminate the player most efficiently.

3.2 Topics

The topics that we utilized in our project are the following:

- Uninformed Search
- Simple Reflex Agent
- Artificial Neural Network
- Genetic Algorithm
- Adversarial Game Playing

3.3 Problem

3.3.1 Relevance

Our project is relevant with a current significant issue, namely the current epidemic, the outbreak of Ebola in Africa. Our project not just aims to inform the player of their own biological capabilities because of the presence of their

immune system, but always promote awareness of the current issue of Ebola, and of the many diseases that run rampant in our world today.

3.3.2 Interestingness

It is interesting because it is an application of our lessons to a hobby relating to entertainment, specifically to video games. A large proportion of students that come into Computer Science wish to create video games, and we believe that this application of our lessons and concepts of AI that we learned both inside and outside of class can actually be applied to things that we perceive as interesting. Our project is an example of the application where our lessons, AI, can be applied to Game Development, and it shows how difficult it is to actually make a good AI that responds well to the user. It gives not just the players but us students further insights about not just Game Development but AI as well.

3.3.3 Ingenuity

It is ingenious because it is an application of multiple AI from different topics to create a specialized AI to solve a specific problem. The use of AI in games is significant in determining the quality of the final output of the game. A good AI translates into a better experience for the player, especially when the AI matches the skill level of the player well. Although AI is commonly used in this field of interest, it is still not as optimized, and this is evident by complaints of players stating that the AI is horrible. Therefore, our project aims to target this issue wherein the AI optimizes itself according to the style and skill of the player.