Gem TD AI

Domingo Martínez Jr.
domingo.martinez01@utrgv.edu

Gabriel Aguiñaga
gabriel.aguinaga01@utrgv.edu

October 2, 2019

Summary of the Proposal

Our plan is to create an artificial intelligence (A.I.) for the strategy game called Gem TD. We hope to achieve this by creating a simplified version of the game and a neural network. The A.I. will be trained using reinforcement learning. We will search for methods to get the A.I. to learn from precalculated probabilities so its productivity and accuracy is increased. If we do not have enough time to optimize the maze design, we may use a predefined maze.

Background

Gem TD[1] is a strategy game categorized as a tower defense. Gem TD was originally created as a custom map for the popular game called Warcraft 3 and has spread to mobile games, web clients, and other games such as DotA 2 and StarCraft 2. This project will focus on the remake of the game for StarCraft 2. The objective of the game is to simultaneously build a maze and place gems to defeat enemy units before they reach the final waypoint. For each wave of enemies, you get to place 5 random gems and keep 1 while the others become rocks for your maze. Gems can be combined into other gems or downgraded. To learn more about the map and maze, see Figures 1 and 2.

We have no knowledge of any existing A.I. for Gem TD. A similar game with regards to randomness that has A.I. is multiplayer Poker[1]. Gem TD is likely the more difficult game for A.I. to play competitively against humans because of the many aspects of the game such as maze design, tower attributes, re-rolls, and upgrades. We are not likely to have enough time to address the problem of maze design, so we will focus primarily on the problem of making the best decisions for keeping gems. To improve the performance of the A.I., we will create a simulator for the game and simplify certain parts with minimal impact to the A.I.'s score in the full-featured game.

[1] https://en.wikipedia.org/wiki/Gem_Tower_Defense

Objectives

The primary objectives are to simulate the game efficiently and develop an artificial intelligence to learn to achieve the best average score on the easy difficulty of damage mode in less than 1000 turns. A game is completed in 51 turns. We probably will not have enough time to train the AI to build its own maze, so a predesigned maze will be used. There are 3 main problems the A.I. can address:

- Decide on what gems to keep and where to keep them
- Decide when to use upgrades
- Decide when to restart a game when a good score is improbable

These secondary objectives are optional as time permits:

- Determine the best maze design for achieving the best average score
- Decide where to place gems in the maze design
- Simulate a strategy to compare against the A.I.
- Develop a graphical user interface for the game

Data and Methods

To achieve the primary objectives, we plan to use reinforcement learning and design a neural network using Python. Data will be obtained from our simulated version of the game. The score is calculated by measuring the total damage done to the final boss, which will always make it from start to end of the map. We can explore the possibilities for enabling the A.I. to learn from precalculated probabilities. To efficiently simulate the game, we may simplify the score contributions from gems into an averaged value. We may simplify to having only instant damage from towers and ignore all enemies except for the final boss. We believe these simplifications will have little impact on surviving the easy difficulty of damage mode. To decide when to restart a game, the A.I. should calculate a probability of achieving a higher than average score while considering the amount of turns remaining. This calculation may be done using probability theory instead of reinforcement learning.

For the secondary objectives, we plan to use SFML[2] or pygame[3] to develop the graphical interface. The maze design shown in Figure 2 will be used and Gem placement will be randomized within the core of the maze unless there is enough time to train the AI. If not using a predefined maze, breadth-first search may be used to route the enemy through the maze.

[2]`https://www.sfml-dev.org`

[3]`https://www.pygame.org`

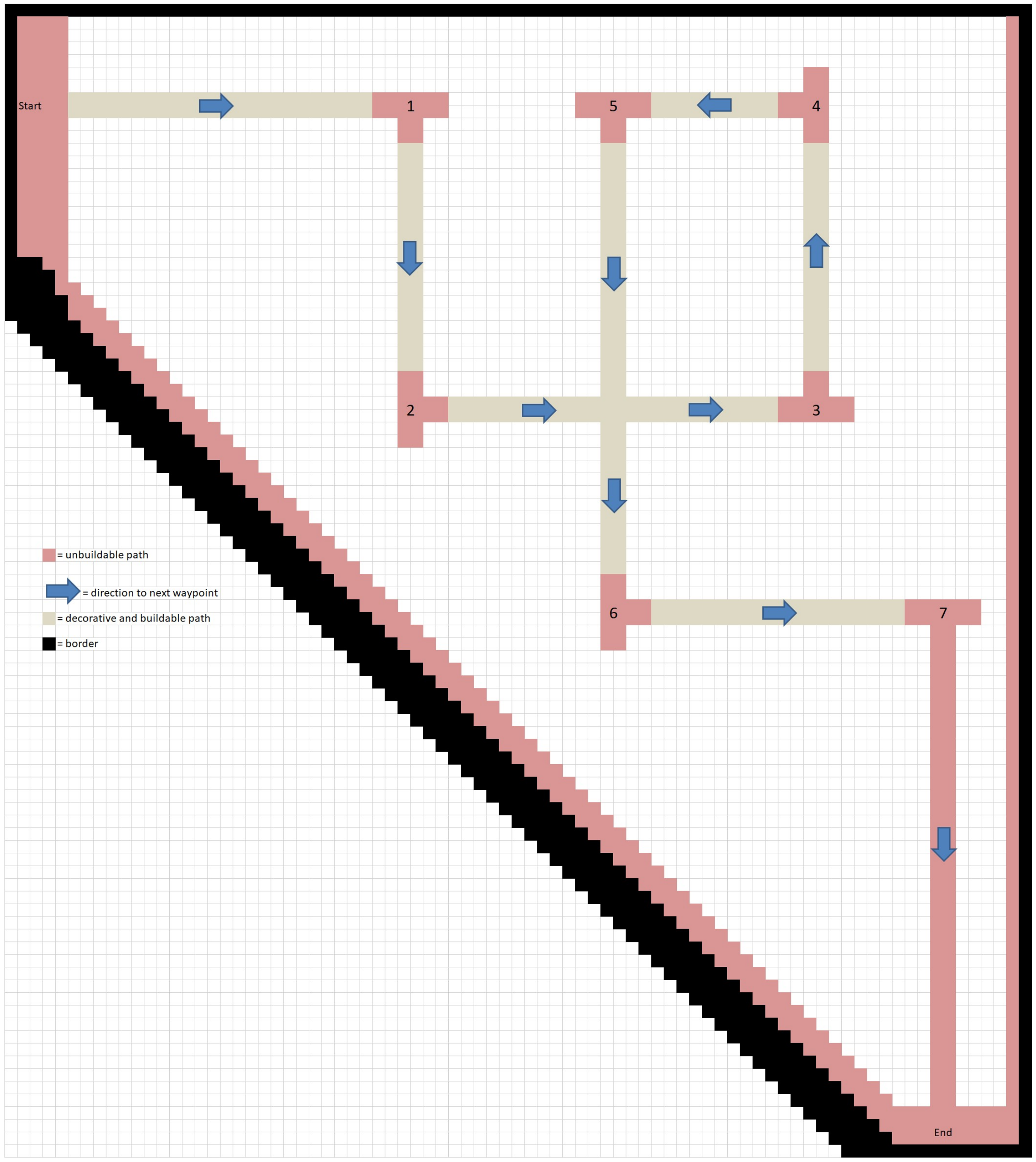

Figure 1: There are 7 waypoints between the start and end. The path the enemy units will take is easy to see when there is no maze. Notice how only the top border can be used to connect to your maze. Notice how you cannot build around the 7th waypoint. You are not allowed to completely block the path to any of the waypoints. It is important to also know there are flying units that ignore the maze. On the next page is an example of a maze that can be completed before the end of the game, and will likely be used by the AI.

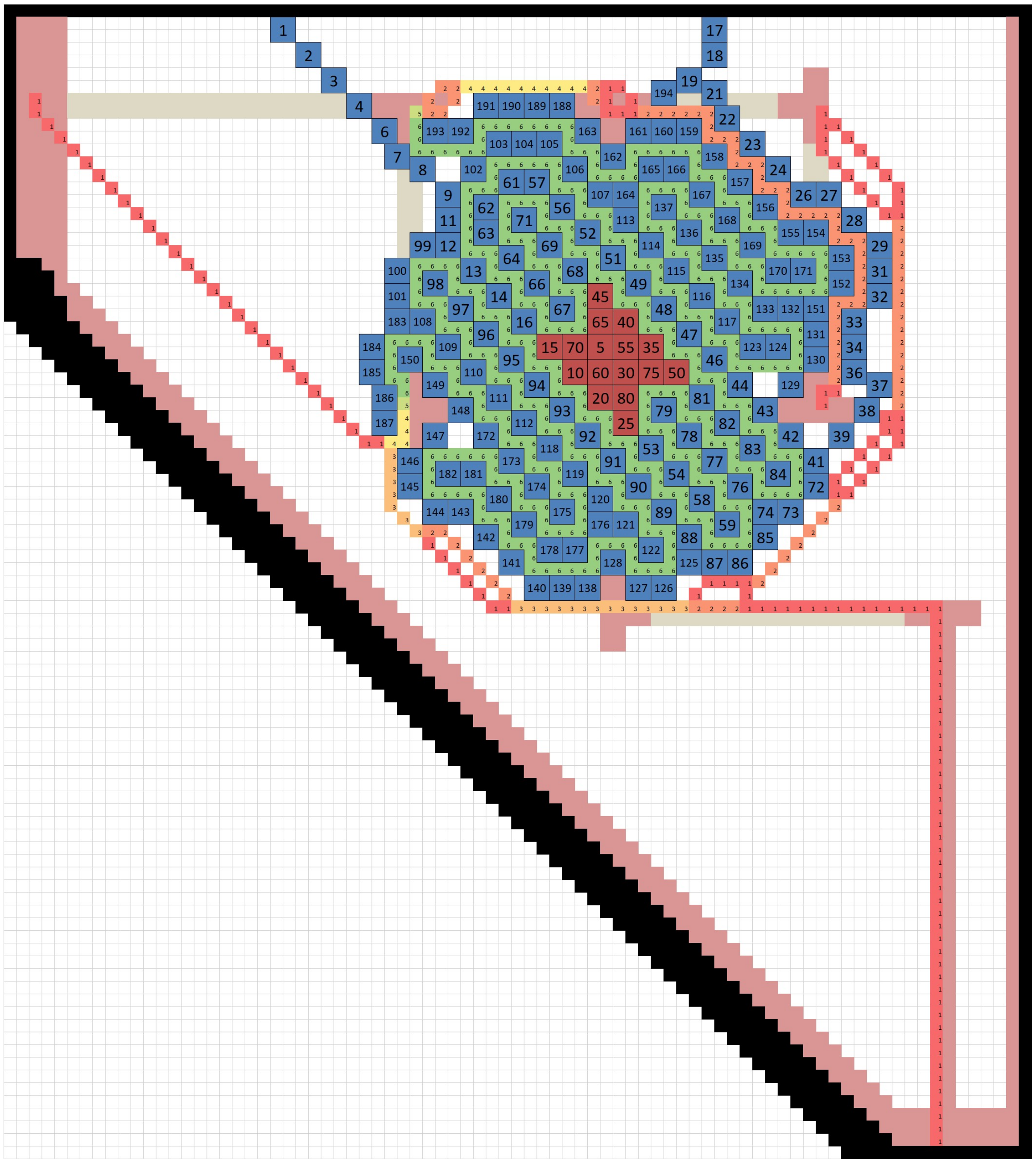

Figure 2: The path of the enemy units is color coded based on the amount of passes through that area, which ranges from 1 to 6 in this example. This maze has 194 rocks/gems in size, and the order of their placements is numbered. The order is not followed exactly in the game. Rocks are intended to take up the blue blocks, and gems are intended to take up the red blocks.

References

[1] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.