Karina Sinha

Game AI 5150

Final Project Writeup

From the start of this project, I wanted my AI to have some different settings in order to mix things up and keep the player interested. I decided I would attempt the stretch goal of “building multiple personas”. To this end, the AI uses rand() to determine two factors: mode and difficulty. Mode refers to whether the AI will implement an offensive or defensive strategy, and difficulty determins whether the AI will be playing on easy, medium, or hard. Both modes have all three difficulty levels. Furthermore, the offensive mode uses rand() to pick a random strategy, and these strategies can sometimes use ran() themselves to vary in their approach.

The AI is structured to determine the mode and difficulty the first time tick() is called and store the results in a member variable. After that, every time tick() is called, it will call the detemined mode’s function. Because the functions depend on elixir inventory to place mobs, it doesn’t matter that they are called every tick. (Note: all of the code was implemented in Controller\_AI\_KarinaSinha.cpp, aside from adding a couple of member variables added in the corresponding header file. All line numbers presented in this writeup will refer to the cpp file).

The defensive strategy (starting on line 112) is fairly straightforward. The AI waits for the player to place a mob and for that mob to get close to the AI’s side before countering with its own mob, which should be blocking the player’s. Which mob the AI places depends on the difficulty and the player’s mob. If the difficulty is easy, the AI places a mob which is weak to the player’s. If the difficulty is medium, it places a mob that is neutral to the player’s, and if hard, it places a mob strong against the player’s. The effectiveness of the mobs was determined through research about the Clash Royale mobs online. I eventually used the Wiki for the game to make the final deceisions, although since ‘swordsman’ isn’t a type of card in the game, I used the stats for knights. For these purposes, Swordsmen beat Archers, Archers beat Giants, and Giants beat Swordsmen.

For the offensive strategy, there was a lot more to consider. Because the AI isn’t just reacting to the player at this point, it needs to generate its own decisions about what to do. As mentioned earlier, the offensive mode generates a strategy with rand(). Rand() returns a number between 0 and 2, and based on that number, the AI will choose one of 3 strategies (for the easy mode, there are actually only 2 stategies, and one of them is just more likely to be selected).

As a way to mix up how the AI makes its decisions, it will sometimes use rand() to generate a number within a strategy and make different decisions based on that. In those instances, the AI will by default spawn in every mob in its strategy, but if the random number is a certain value, it will only spawn a couple of them, thereby using less elixir and needing to wait for a longer period of time to reset to ts default. An example of this can be seen between lines 261 and 298.

In addition to the stretch goal of “building multiple personas”, I also attempted the “AI says things via cout” stretch goal. However, I couldn’t really figure out how to keep the messages shorter than a few words. Furthermore, due to the fact that tick() calls each function over and over again, the print statements were repetitive and blocked out the important information (such as the other statements about the towers attacking and such). Therefore I commented out most of the AI’s dialogue. Example can be seen in tick() when the strategies are called (lines 68 and 75) as well as in startDefensive() (lines 92 though 110) and selectOffensive() (lines 212, 245, and 319). Some print statements are left in as flavor text, and they do not occur repetitively.

The code has comments scattered throughtout the code to explain its decision making process, so looking through it should further explain how it is structured.