Design Decisions and Data Structure Justification

Ryan Boucher

October 26, 2018

1 Java Class: PlantPieces

This class allows for text-based representation of the pieces on that will be used in the game. This is primarily used for milestone one, in which we use a text-based display of the basics of the game.

There are no significant data-structures to discuss in this class.

2 Java Class: Piece

This class models pieces used in this game, and includes information on individual piece health, damage, cost, and both the name, and short-name (char) that is used to represent the piece. Pieces will exists as objects in this design, and as such, it makes logical sense to model them here.

There are no significant data-structures to discuss in this class.

3 Java Class: Gameboard

This class models the board that the Plants VS Zombies game runs on. Additionally, it holds information on the amount of money that the user currently has, as well as the limit on the number of zombies that will occur on the board (a rough version of difficulty). As the amount of money that the current player has is being modelled in this class, the "Gameboard" is synonymous with "Level" in this design, with each new level having a new gameboard, thus resetting money and stats.

Data Structure: Board : The board in this class has been modelled as a two-dimensional array, to allow for the use of rows, and columns. Two dimensional arrays are fast in regards to their access of information, and as the board is small (8 columns by 5 rows), the game has lost little in regards to efficency when it iterates over the board.

4 Java Class: Coordinate

This class models coordinates that will be used to identify locations on the board, through columns (X values) and rows (Y values). This is necessary to allow for easy identification of where pieces are in the game.

There are no significant data-structures to discuss in this class.

5 Java Class: Square

This class models the individual squares that make up the game board. It is used when adding and removing pieces, and will identify if a square is currently being occupied or not. This class contains the logic for adding and removing pieces on the board, and is integral for the game.

There are no significant data-structures to discuss in this class.

6 Java Class: Main

This class models the game itself, and runs turns until either the player loses, wins, or quits. The class initializes with a small ascii splash-screen, allows the player to start a game, and then repeatedly runs startGame() until the game is finished. At the end of each turn, the runGame() method is used to add money to the player's money count (held in the gameboard), and start the CPU's turn (moving and adding zombies, and proceeding with an attack if available).

There are no significant data-structures to discuss in this class.