Sean Howley

Dan Burns

Ryan Green

Connor Green

**Tower Defense: User Manual**

**Overview:**

Tower Defense is a strategy based game which consists of enemies traversing through the path on a map in order to attack the user’s home base at the end of the path; once enough enemies reach it, the base is destroyed and the user loses. In order to prevent the enemies from reaching the base the user is able to place towers- which damage the enemy’s health- anywhere on the map (besides the path that the enemies are following) in order to strategically destroy the enemies. The different towers that the user is able to choose from consist of a basic tower that is not very strong and damages the enemies slowly, a stronger tower that incurs more damage to the enemy, and tower that does not damage the enemies but slows them down which can be strategically paired with other towers to maximize damage. In order to utilize these towers the user must have enough money (the cost of the towers mentioned increases respectively)- they begin with $500 and gain money for each enemy destroyed- and if they do they are provided with a drag-and-drop feature along with a menu that displays each tower and its respective cost. The game provides the user with only one map, but each time the user completes a level without having their base destroyed, the new wave of enemies becomes more diverse (there are different enemies with different speeds and health) and the number of enemies in each wave gets larger. Along with the tower menu at the bottom of the screen, the user’s total money is displayed and the points they earn is based on the amount of money that they spend by the end of the game in order to encourage the user to be as efficient as possible with the towers that they place.

**User-Controls:**

* Start Screen:
  + Upon running the program, the user is provided with a start screen that gives the option to either click a button to start the game, or click a button to see a local leaderboard which keeps track of the scores of people that have played on the local machine.
* Placing Towers:
  + The user is given 15 seconds to place towers, then the first wave of enemies are automatically sent out. Once a level is complete another 15 second wait period is given to place towers before the enemies enter the map.
    - However, the user is still able to place towers when the enemies are in motion.
  + If the user tries to place the towers in an invalid place (ie. on the path or in the start menu), they are notified that it was an invalid placement and it is not placed.
* Easter Eggs:
  + Play around with some keys and you might find some useful cheats!

**Specs:**

* Health Bars:
  + Enemies:
    - Red health bars are displayed above the enemies’ in order to show how much health they have left, and decrease in real-time as they get attacked.
  + Base:
    - The user’s base also has a health bar underneath it that decreases in real-time as enemies breach it in order to show the user how close they are to losing.

**Creator’s Guide:**

This program is implemented using the python graphics library called pygame. After an immense amount of research, we decided that the most efficient way to optimize the speed of the game and prevent lagging was to utilize pygame’s sprites. In order to understand how the game functions from the programmer’s perspective you must first understand how sprites function. Sprites essentially serve as a base class that is inherited from while dealing with graphics that are in motion; these are particularly useful because each instantiation inherits from this base class which contains an update function that is made to efficiently erase and redraw the class’s object without having to erase the entire screen- this can be utilized to efficiently simulate motion. Further, the equivalent of making a list of these objects in order to have them behave as a group (since we need multiple enemies, towers, etc), there is a pygame feature called groups. Groups can hold as many sprites necessary and are able to invoke each sprite’s update under the same conditions in order to allow them to behave in synchrony. We utilized these features while creating an enemy sprite which controls enemy’s movement, speed, and health, and a tower sprite which controls the tower’s location, strength, and attack range. Inside of the enemy sprite’s update function we controlled the movement and speed by incrementing the coordinates by a factor of the velocities and providing the correct bounds of the path such that the velocities change signs to perform turns. The tower sprite’s update is also similar in this sense because although it is not always moving, when it is being dragged it is changing according to the mouse location and in order to make sure that it is not placed in an invalid area it checks for collision with the path. While all of these sprite instantiations and groups are updating and interacting in the main function, the menu at the bottom of the screen is simply represented with constant images and the time and money are reprinted to the screen whenever they change.