CS179N Final Project - Tower Titans

Team Members: Augustine Ayoub, James Krejci, Michael Tin, Yuze Fu

Game Introduction

"Tower Titans" is a 2D, top-down tower defense game where the player will strategically defend a map against waves of enemies.

- The main objective of the game is to place towers to defeat the enemies before they reach the end of the map.
- Over the course of the game, the player will earn gold for defeating enemies and completing levels.
- The gold can be used to upgrade towers with perks to make them more effective, as well as purchase walls which will slow the enemies down.

Game Description

The "Tower Titans" game features 50 unique levels of gameplay. The player can choose between Beginner, Intermediate, and Impossible difficulty settings. The different difficulty settings control what types of enemies are spawned.

The player can place down towers on any of the green tiles and walls on the beige path. Towers can be upgraded up to two tiers: Tier 1 is a cyan color tower, while Tier 2 is a purple color tower. The player can choose two upgrade paths to reach Tier 1 and Tier 2. The left path increases fire rate, while the right path increases range and damage.

The enemies vary in color, with green slimes being the easiest due to their moderate health and speed, followed by red and purple slimes with their varying intermediate health and movement speeds, and finally orange slimes presenting the greatest challenge with their high health and fast movement. The game also features a "Boss Enemy" which spawns every five levels.

The player can earn gold (the currency of the game) by defeating enemies and surviving waves. Our game rewards gold based on the type of enemy slain (so if a green slime starts off with 2 HP, the player earns 2 HP) Surviving levels reward the player with 100 gold to boost tower upgrades and buy more towers.

The game also includes a "Save Game" feature, which allows the player to save their progress and come back to the game at a later time. This feature saves the current state, difficulty setting, and level progression of the game.

Game Implementation

The “Tower Titans” game is implemented using the Unity 6000.0.45f1 engine, which allows us to create modular systems.
The gameplay is driven by C# scripts that handle the logic and the user interface. The following is a description of the main systems/logic in the game.

1. Enemy Prefabs/Enemy Spawner System
    a. There are four types of regular enemies in the game.
        i. Tier 0: Health = 2, Move Speed = 2
        ii. Tier 1: Health = 5, Move Speed = 3
        iii. Tier 2: Health = 10, Move Speed = 4
        iv. Tier 3: Health = 25, Move Speed = 5
    b. There is also a Boss Enemy that spawns every five levels (Level 5, Level 10, etc.) and doubles its health every time it spawns. On boss enemy levels, the boss enemy spawns after all the regular enemies have spawned.
        i. Boss Enemy: Health = 100 (starting health), Move Speed = 2
    c. The sequence of the enemies for all levels is predefined through a 2D array (array of arrays). An individual array represents the sequence of the enemies that will spawn for that particular level.
    d. The user’s difficulty selection controls the types of enemies that they will interact with in the game. The first enemy is the “base type”, and the second enemy is the “upgraded type”.
        i. Easy Difficulty: Tier 0 and Tier 1
        ii. Intermediate Difficulty: Tier 1 and Tier 2
        iii. Impossible Difficulty: Tier 2 and Tier 3
    e. The first level starts off with three base enemies (Tier 0/1/2). The next level makes the last two enemies an upgraded type (Tier 1/2/3), and then the level after that adds three more base enemies (Tier 0/1/2).
    f. This pattern alternates to create the game sequence for the rest of the game. That is, even levels upgrade the last two enemies in the array, and odd levels add on three more base enemies.
2. Enemy Pathfinding System
    a. The path is a list of points in the map. The enemy set its linear velocity to be the movement speed times the normalized direction vector(path point position - enemy position) .normalized()
    b. When the enemy’s distance to a path point is below 0.1 float, the enemy chooses the next path point in the vector.

3. Arrow/Tower Rotation Logic
    a. We take the normalized direction vector = (target position - bullet position).normalized() first. Then, the angle we should rotate the arrow to can be calculated by using arctan(y/x), where y is the y component of the normalized direction vector, and x is the x component of the normalized direction vector.
4. Gold Rewarding System
    a. The player earns gold over the course of the game.
    b. The gold is rewarded when the player kills an enemy.
        i. The gold rewarded is the enemy's original health (2, 5, 10, etc).
    c. The gold is also rewarded at the end of each level.
        i. The gold rewarded is 100.
    d. The gold that the player earns can be spent on buying towers, buying walls, or upgrading towers.
5. Wall / Tower Purchase System (Integration with Gold Rewarding System)
    a. Wall Manager/Tower purchase manager/Tower upgrade manager calls the gold API to get the current player's gold amount. If the gold amount is greater than or equal to the cost, the managers authorize the purchase and call the gold API that deducts the cost from the player's gold amount.
        i. Tower price: 50g
            1. Tier 1 upgrade from either upgrade path: 10g
            2. Tier 2 upgrade from either upgrade path: 20g
        ii. Wall price: 20g
    b. The tower upgrade manager allows the player to sell their towers. When the player sells their tower, the tower upgrade manager calls the gold API to increase the player's gold by 40 gold.
6. Tower Upgrade Logic
    a. Each perk is a node in a linked list, with a pointer pointing to the next perk(node). Each perk(node)'s data includes modifiers to attack range, attack damage, fire rate, and cost to purchase the perk.
    b. Based on each node's pointer, the tower upgrade manager generates the UI to show the player which perk is available for purchase. The tower upgrade manager applies the appropriate modifier to the tower when the perk is purchased. The tower upgrade manager uses cost data of each node to determine if the player can afford the upgrade, and deducts the cost from the player when the player purchases the perk.

7. Screen Shake Effect
    a. We record the original camera location. (x,y,0)
    b. We generate a random point within the unit circle (radius of 1) in 2D.
    c. We pad the 2D vector of random points' z value to be 0, and make it a 3D vector
    d. We set the current camera location to be the sum of a random point(now a 3d vector) and the original camera location.
    e. We set the camera location back to the original camera location.
    f. We repeat this process until the camera shake duration ends.

Game Development Post Mortem

All tasks planned out at the beginning of the quarter were completed.

1. Save Game: The player is able to save the progress of the current game into a save file.
2. Load Game: The player is able to load a save file in the main menu and continue where they left off previously.
3. Display Score: The player will be able to see their score and level at the top right of the screen.
4. Place Tower: The player is able to purchase and place a tower anywhere on the map grid before the beginning of each level.
5. Lose Health: The player's health bar will decrease proportionally to the number of enemies that make it to the end of the map grid.
6. Earn Gold: The player earns gold when defeating enemies as well as at the end of each level.
7. Change Difficulty: The player is able to pick a difficulty (easy, medium, hard) at the beginning of the game and change the difficulty during the game.
8. Upgrade Towers: The player is able to use their gold to upgrade the abilities of their towers at the end of each level.
9. Place Walls: The player is able to purchase and place walls anywhere on the map grid before the beginning of each level.
10. Boss Challenge: At certain points in the game, the player will face "boss level" enemies with increased health as a challenge.
11. Frontend: The game interface and environment will have a cohesive visual style, including various graphical assets that match the game's intended theme.

There were no incomplete tasks.

Scrum worked efficiently for us.

- We split up tasks evenly through the quarter and held two meetings each week, one on Monday during lab and the other on Friday.
- If our team members became busy due to unforeseen circumstances or midterms, they communicated effectively to the whole group.
- This allowed us to help those in need and to finish the project on time.
- Overall, it was calming to have scrum meetings twice a week so we know who needs help shipping their features and plan ahead for the next assignments presented at the next upcoming scrum.

In terms of what we could improve...

- We would try to get the core functionality of the game working by mid quarter so that we can spend more time designing and adding custom features to make our game more unique.
- We would like to add music and sound effects, as well as a greater variety of towers, enemies, and maps that would help improve the game greatly.
- It would also be interesting to see if we could add different types of "tower defense" objects to make the game strategy more diverse.

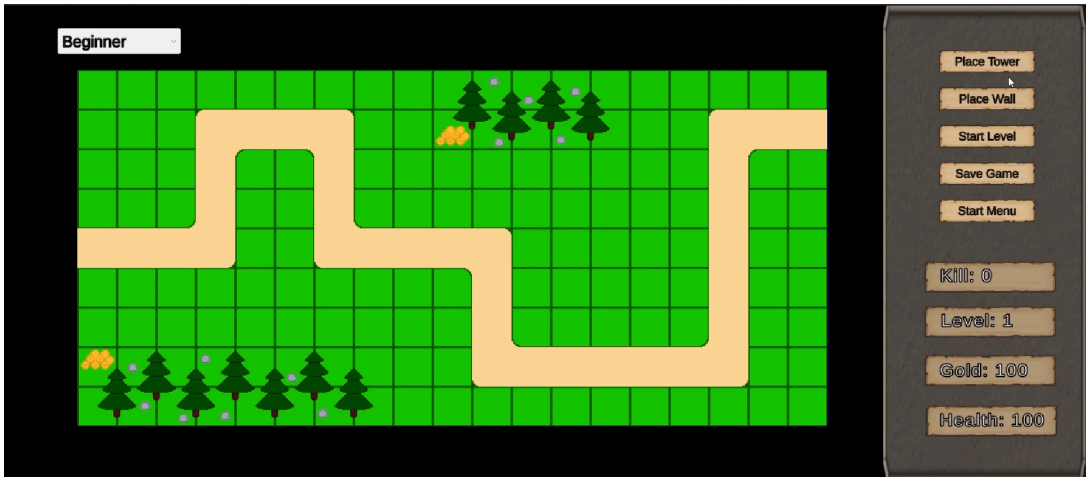
In terms of challenges that we faced...

- It was initially difficult to learn Unity and C# for the first few weeks of the quarter, since we were unfamiliar with the development environment.
- We had to be proactive in coordinating schedules and consistently finding time to work on the project each week, as that was not always a given.
- We experienced some various technical challenges throughout the development process. This included the following:
    - Making sure the different layers interacted properly (i.e. the tower does not shoot another tower or object that is not an enemy)
    - Making sure that the objects in the scene were properly deleted upon being deleted (i.e. a tower kills an enemy)
    - Making sure that the different systems (developed by different team members) interacted properly
- We were eventually able to properly debug all these bugs and errors to make a fully functional game.

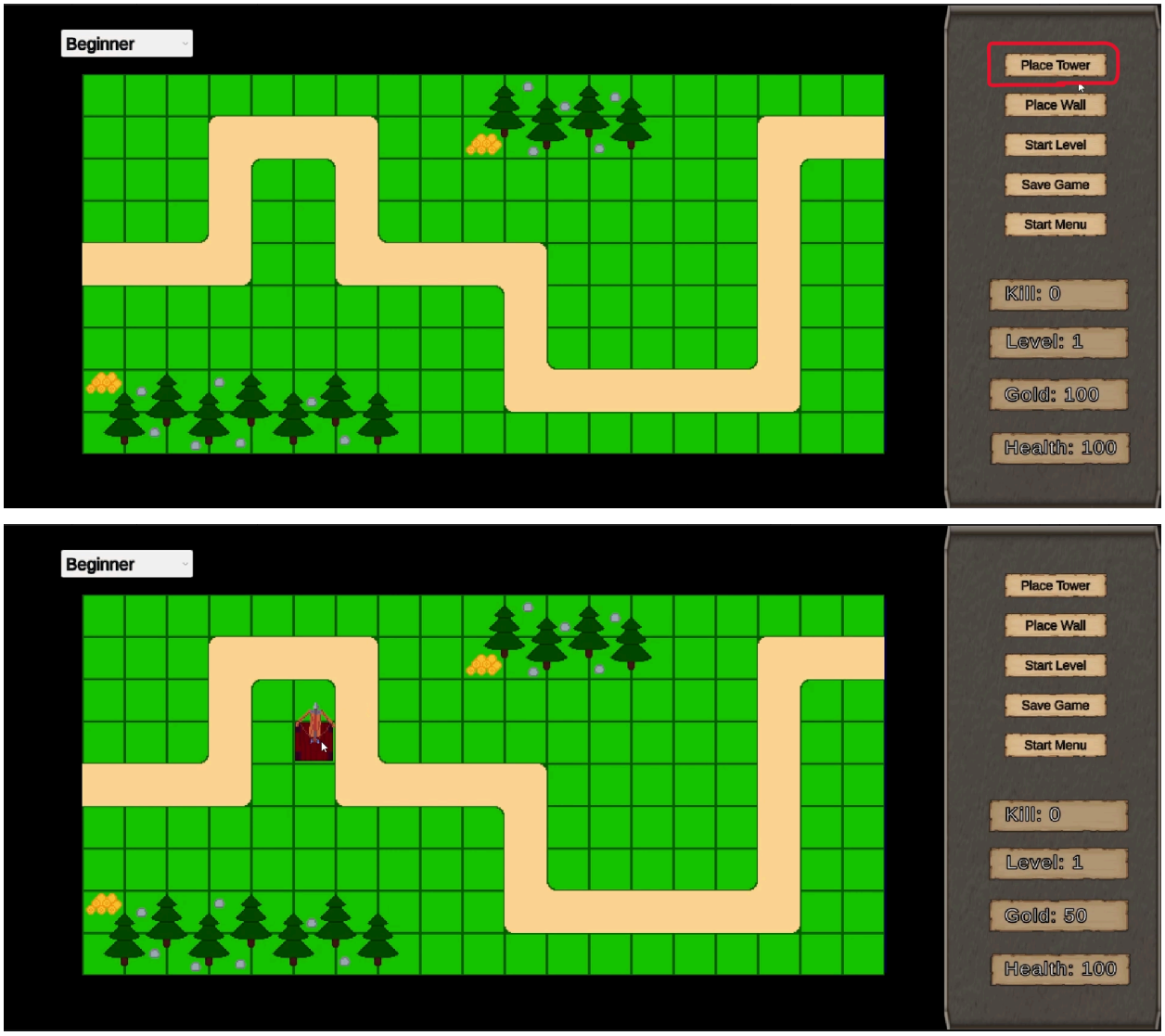
Game Demonstration

Upon starting the game, the player is taken to the main menu.

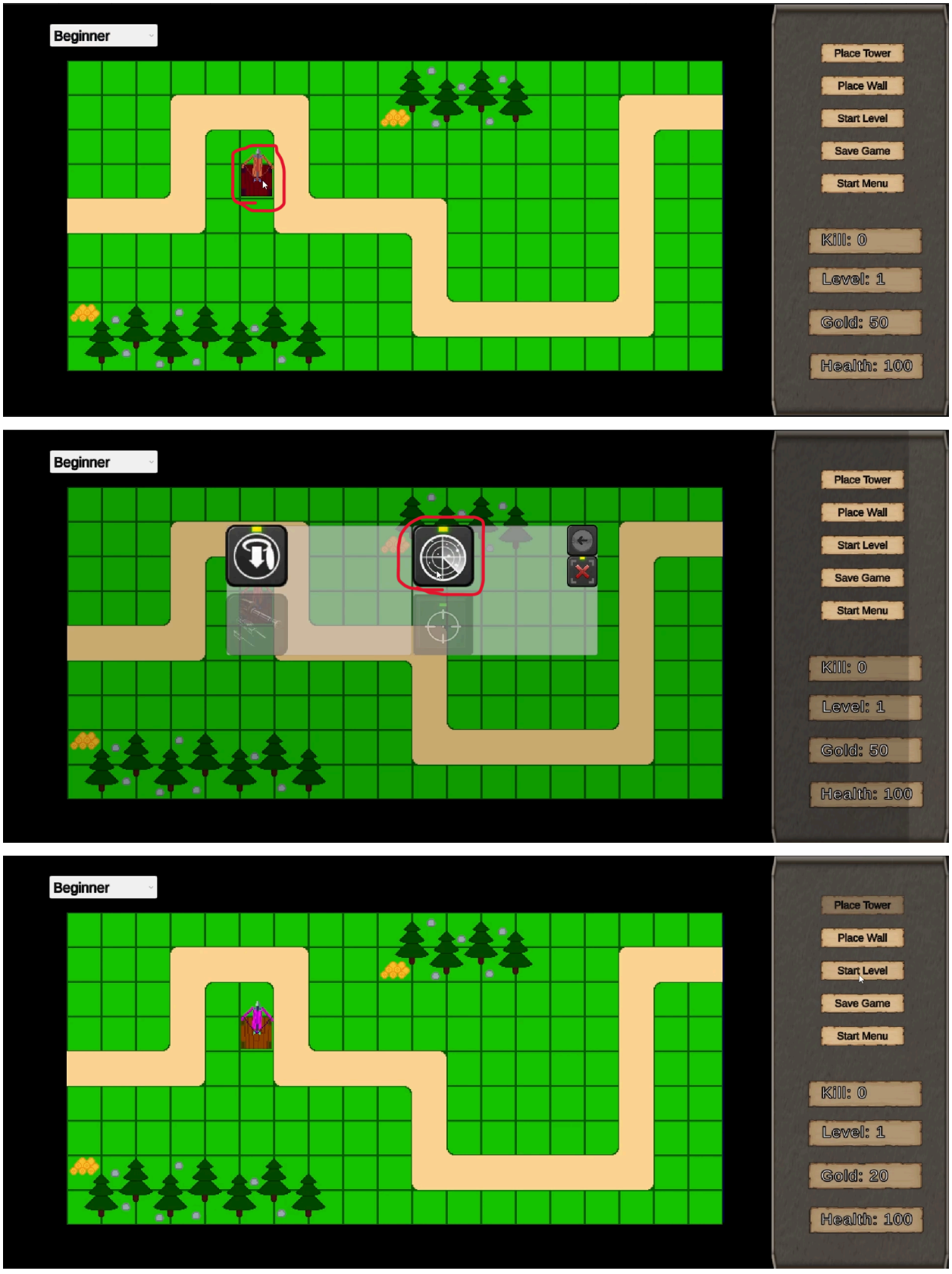
“How To Play” gives a brief overview of the game.



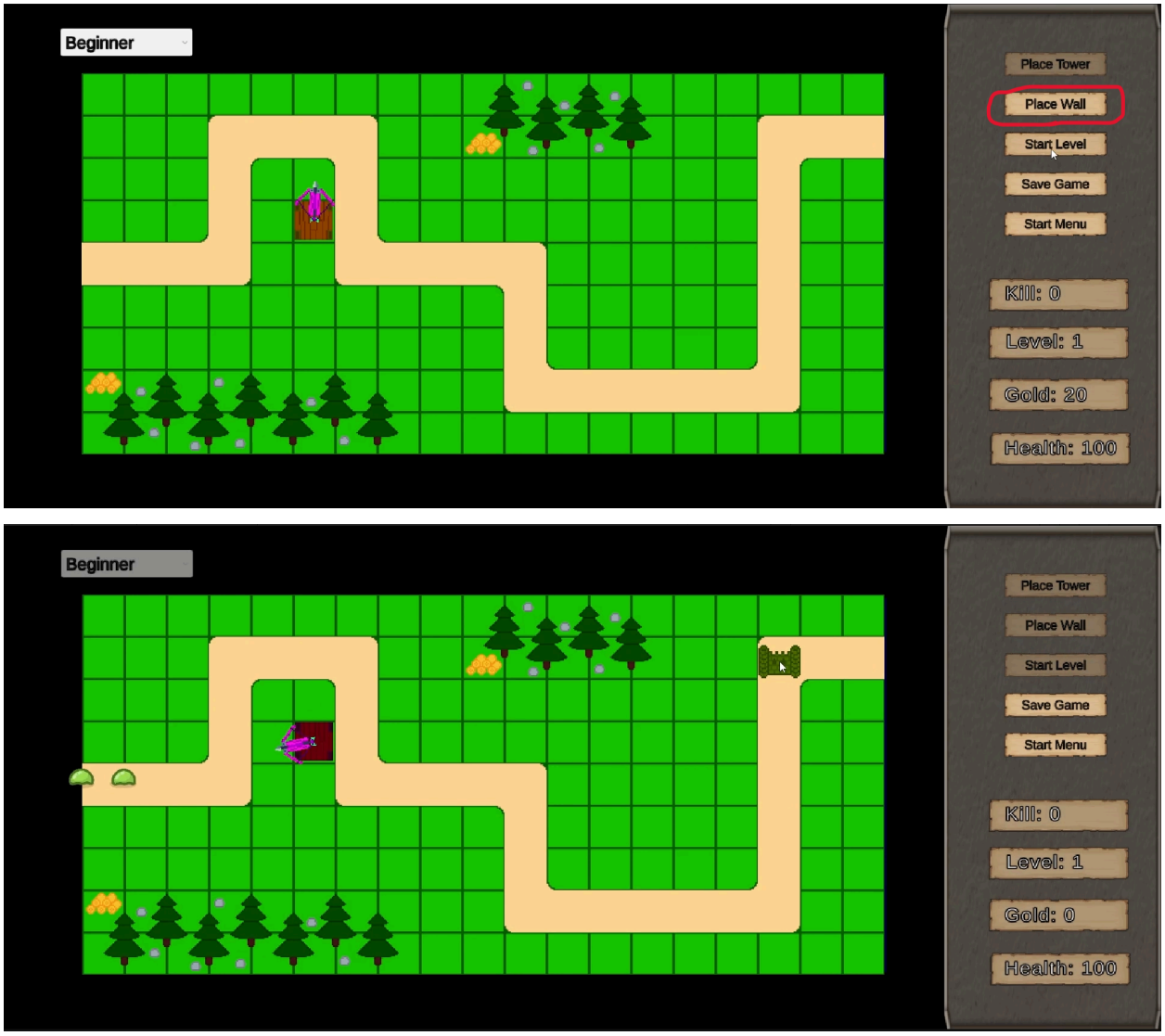
“Start Game” will take the player to the map.



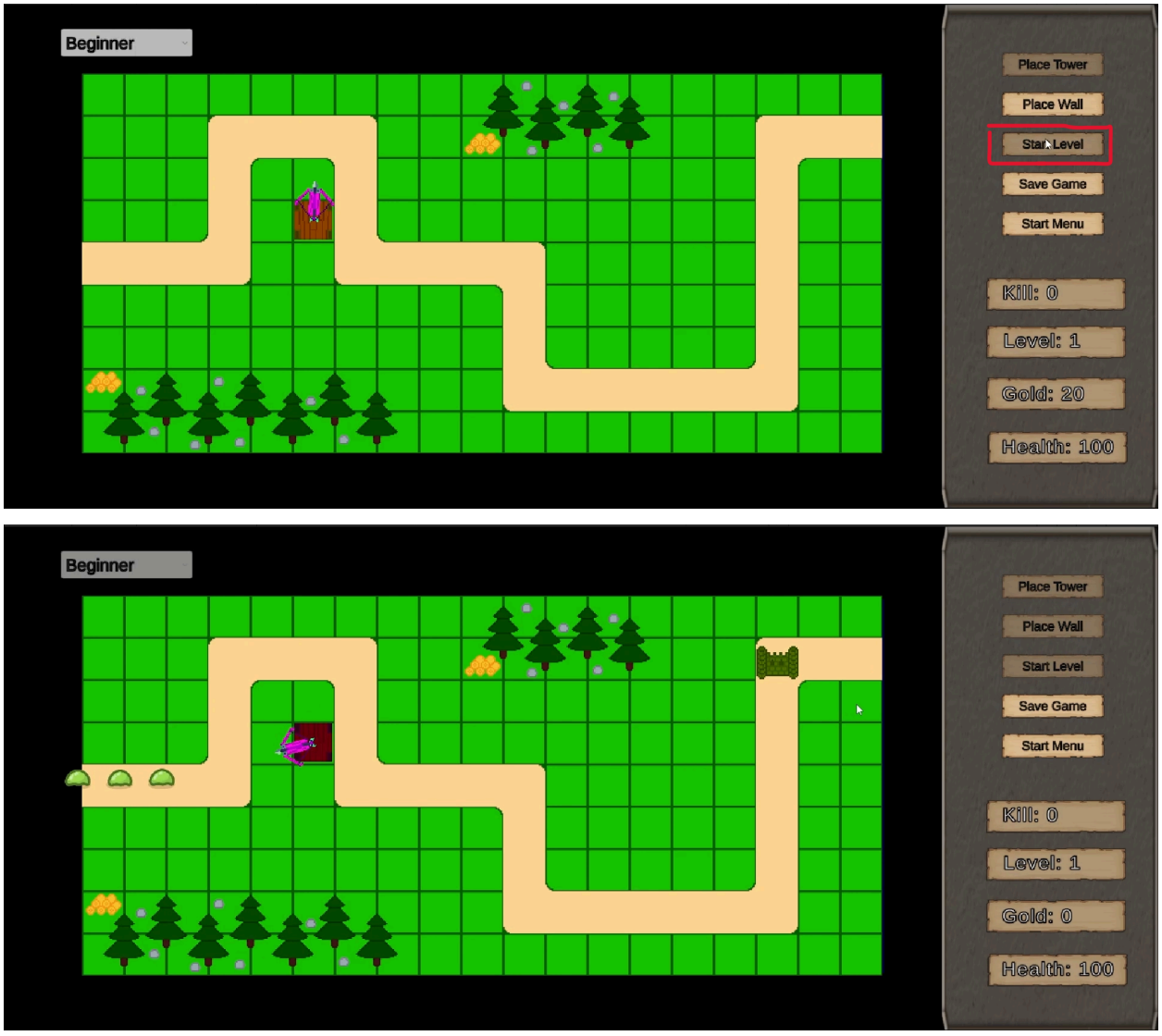
By selecting the “Place Tower” button and then selecting an empty space on the map, the player can place a tower.



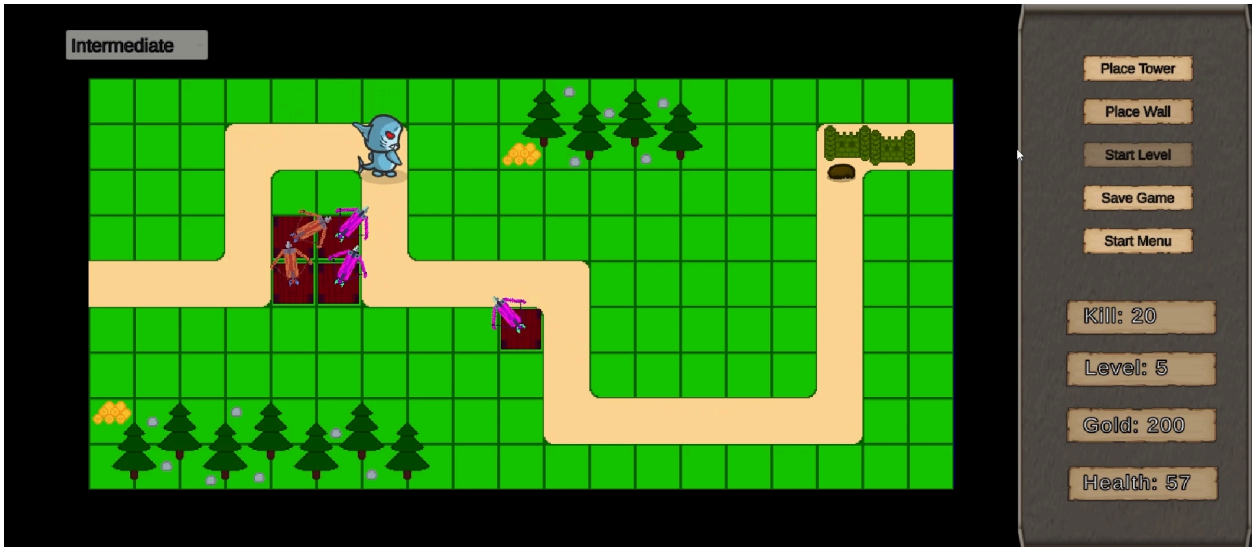
By selecting a placed tower, then selecting upgrades, the tower can be upgraded.



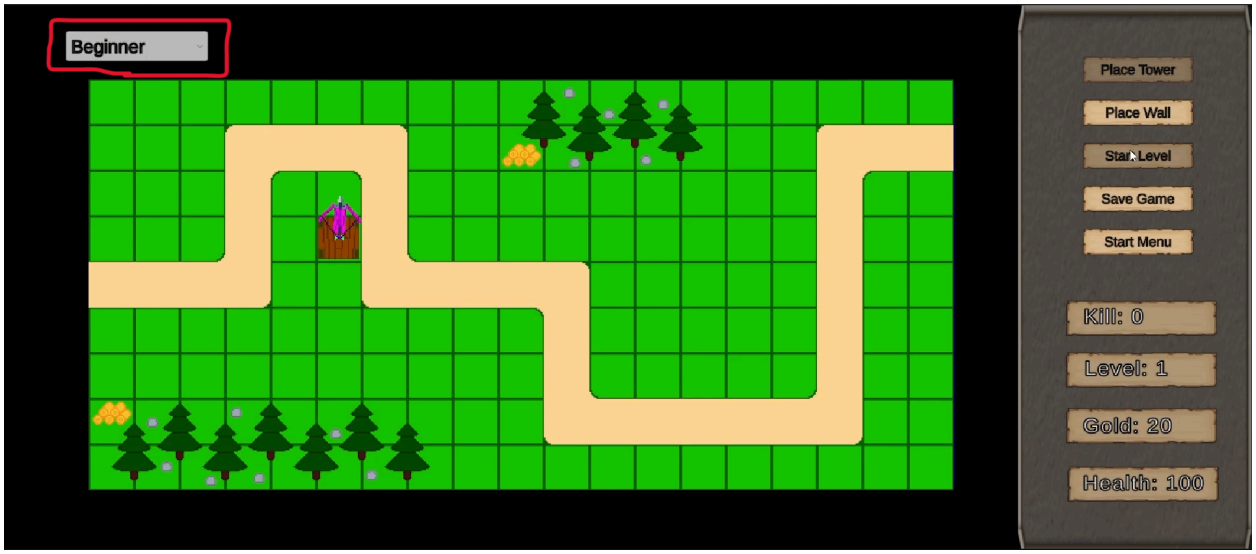
By selecting “Place Wall” and then selecting a point on the path, the player can place a wall that will damage enemies that run into it.

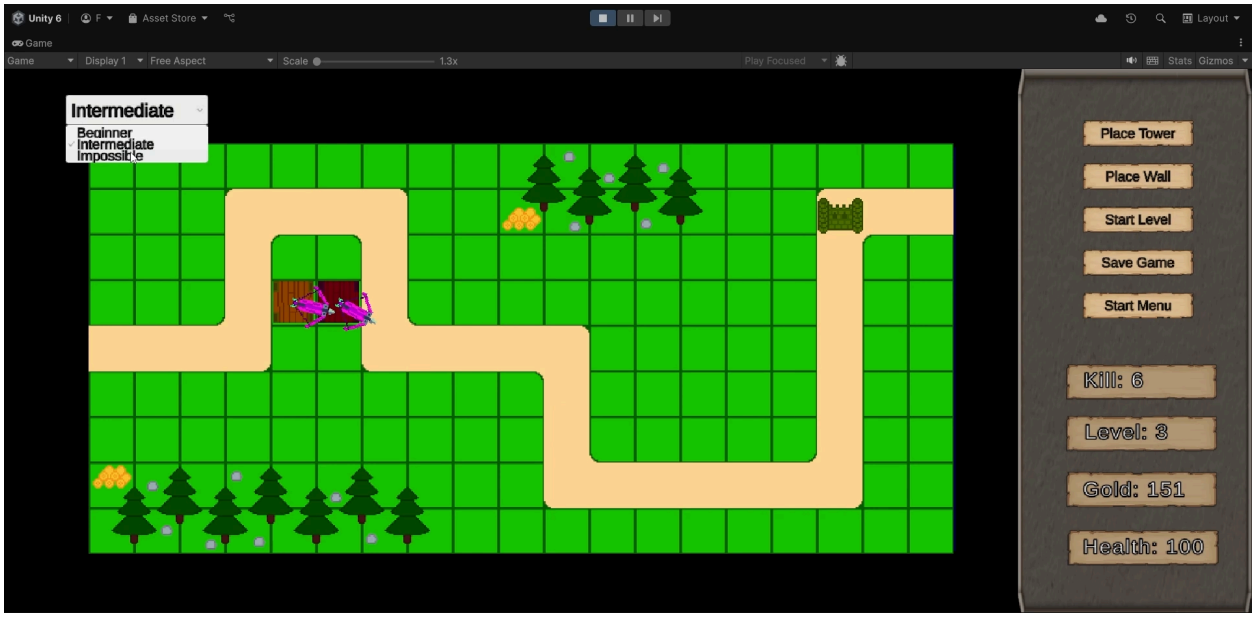


By selecting “Start Level,” the player can begin the game. Slimes will follow the set path while being shot at by towers.

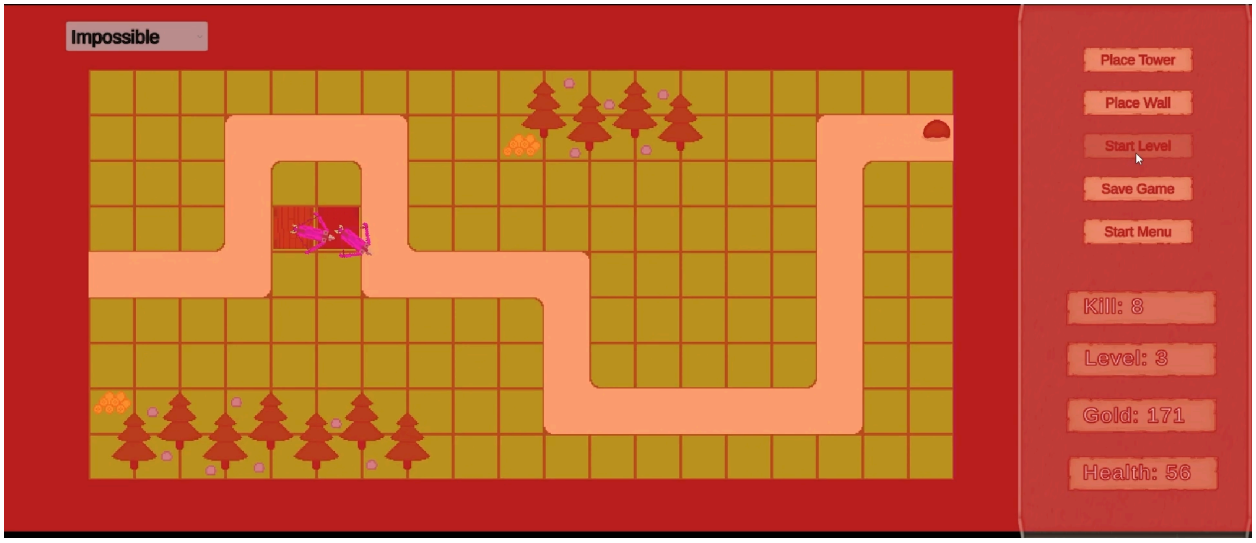


Every five levels, a larger boss enemy with more health will appear.





By selecting the difficulty selector in the top left corner, the player can adjust the difficulty level between rounds, which affects the health and speed of enemies.



When enemies get past the player's defenses to the other side of the screen, the screen flashes red, and the player's Health goes down.

When player Health hits zero, the player has lost and Game Over appears on screen.

By selecting “Save Game,” a player can save their progress for later. A player can enter a previously saved game by pressing “Load Game” from the main menu.