Name: David Bush Mark _____/50

1. Brief introduction _/3

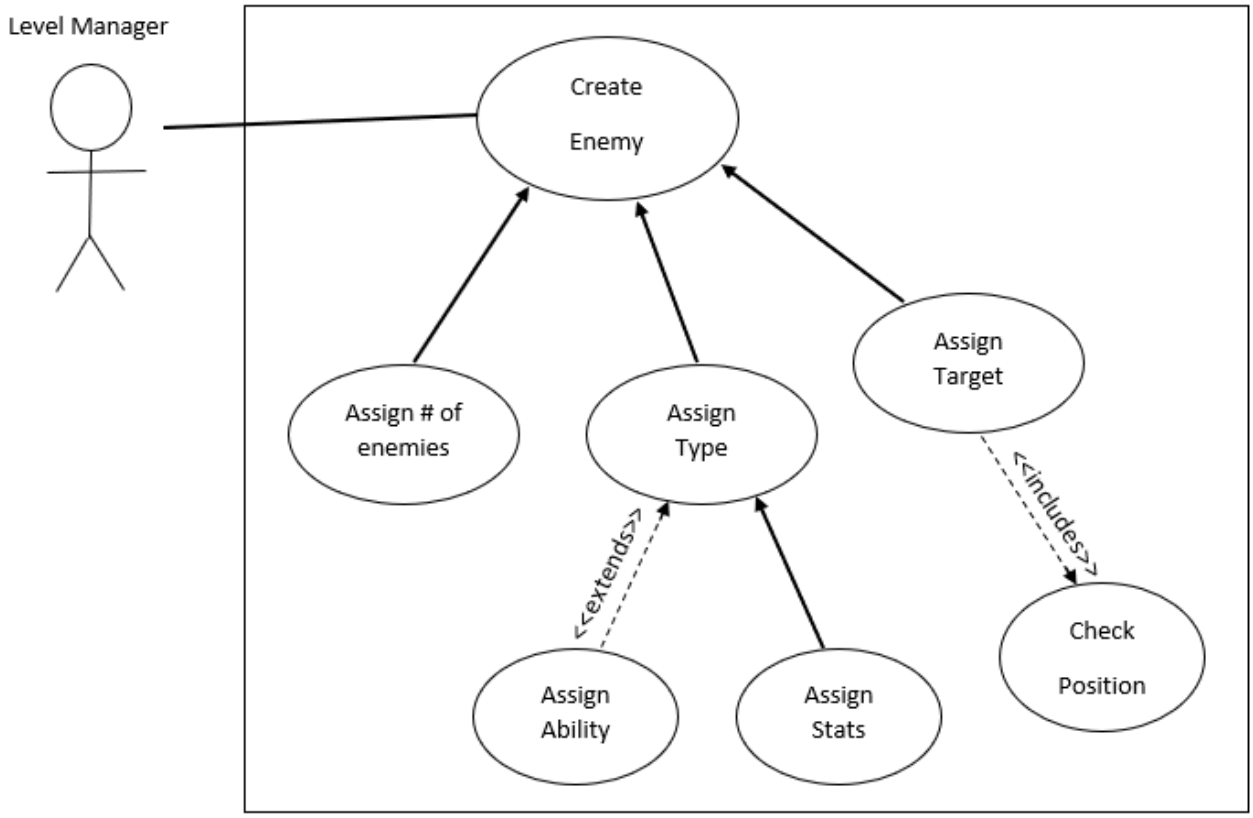
My feature for the Wonky Wizards tower defense game will be creating the enemy characters and managing their interactions.

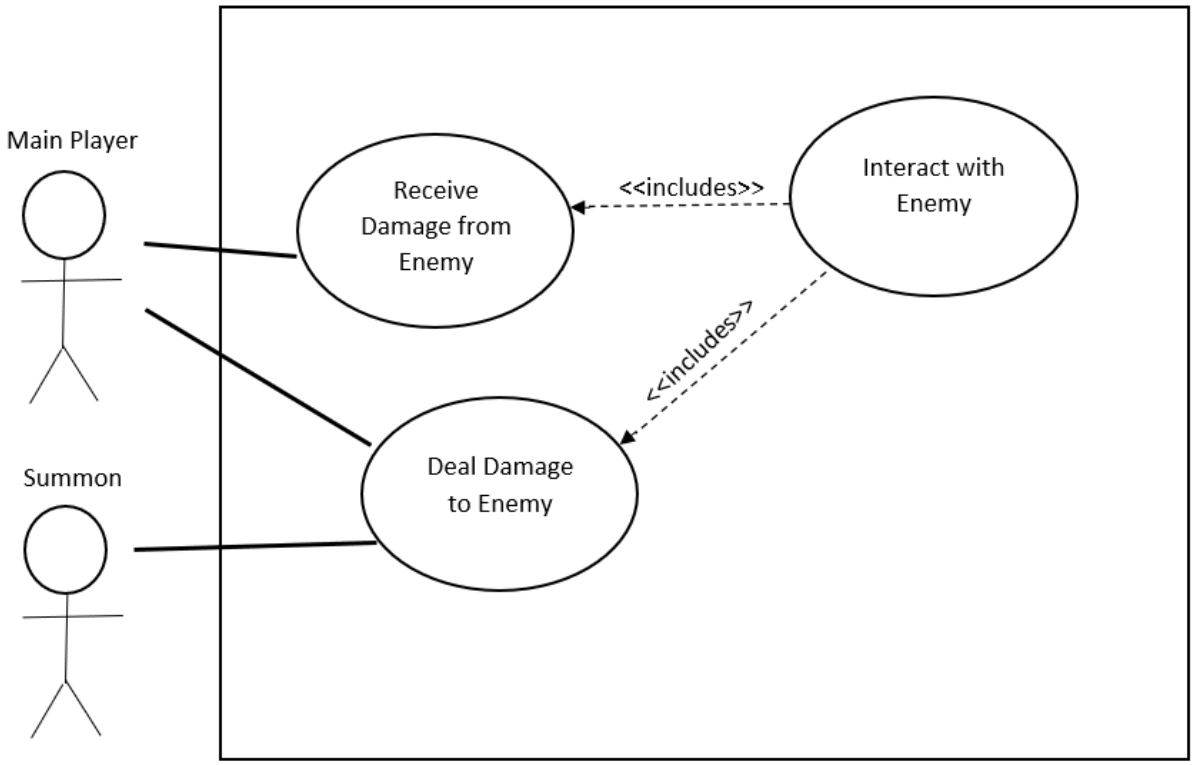
At the start of each round, a wave of non-player characters (NPCs) will be spawned in and move towards the goal target. My job will be creating the enemy NPCs and keeping track of their attributes which include hit points, speed, damage, range, and abilities. I also need to control the NPC's movement, pathfinding, and targeting. I will need to make sure the enemies can't move through the summon objects that are impassible and ensure there's a clear path to the target using a pathfinding algorithm such as A*. I'm also responsible for the enemy characters' interactions which the main player, summons, and goal. I will need to keep track of each enemies' damage received, damage output, position, and ability status.

Additionally, I will be working on the demo mode for our game which will run without user input and demonstrate how to complete the level as well as a failure case.

2. Use case diagram with scenario _14

Use Case Diagrams





Scenarios

Scenario 1 (1st Use Case Diagram)

Name: Create Enemy

Summary: The level manager oversees when a new enemy needs to be spawned into the game.

Actors: Level Manager

Preconditions: The game round has started.

Basic sequence:

Step 1: Accept the number of enemies required and their corresponding types.

Step 2: Assign the stats of the enemies and abilities if they have one from their pre-set classes.

Step 3: Retrieve the goal's position and update the enemies target to the goal.

Step 4: The waves of enemy goblins are now ready to commence!

Exceptions:

Step 1: Check if the enemy type has an ability to use.

Step 2: There has been no input for number of enemies.

Step 3: There has been no input for type of enemy.

Step 4: The target position is not found.

Post conditions: No more enemies are required.

Priority: 1*

ID: CE1

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Scenario 2 (2nd Use Case Diagram)
Name: Interact with Enemy
Summary: The summons and main player need to interact with the enemy characters.
Actors: Main Player, Summons
Preconditions: Enemies have been created.
Basic sequence:

Step 1: Check the status of the enemy.
Step 2: Check if the main player or summon has dealt damage to the enemy and apply that damage.
Step 3: Check to see if the enemy has dealt damage to the main player and send that damage to the main player.

Exceptions:

Step 1: Enemy is dealt lethal damage.
Step 2: Player is dealt lethal damage.

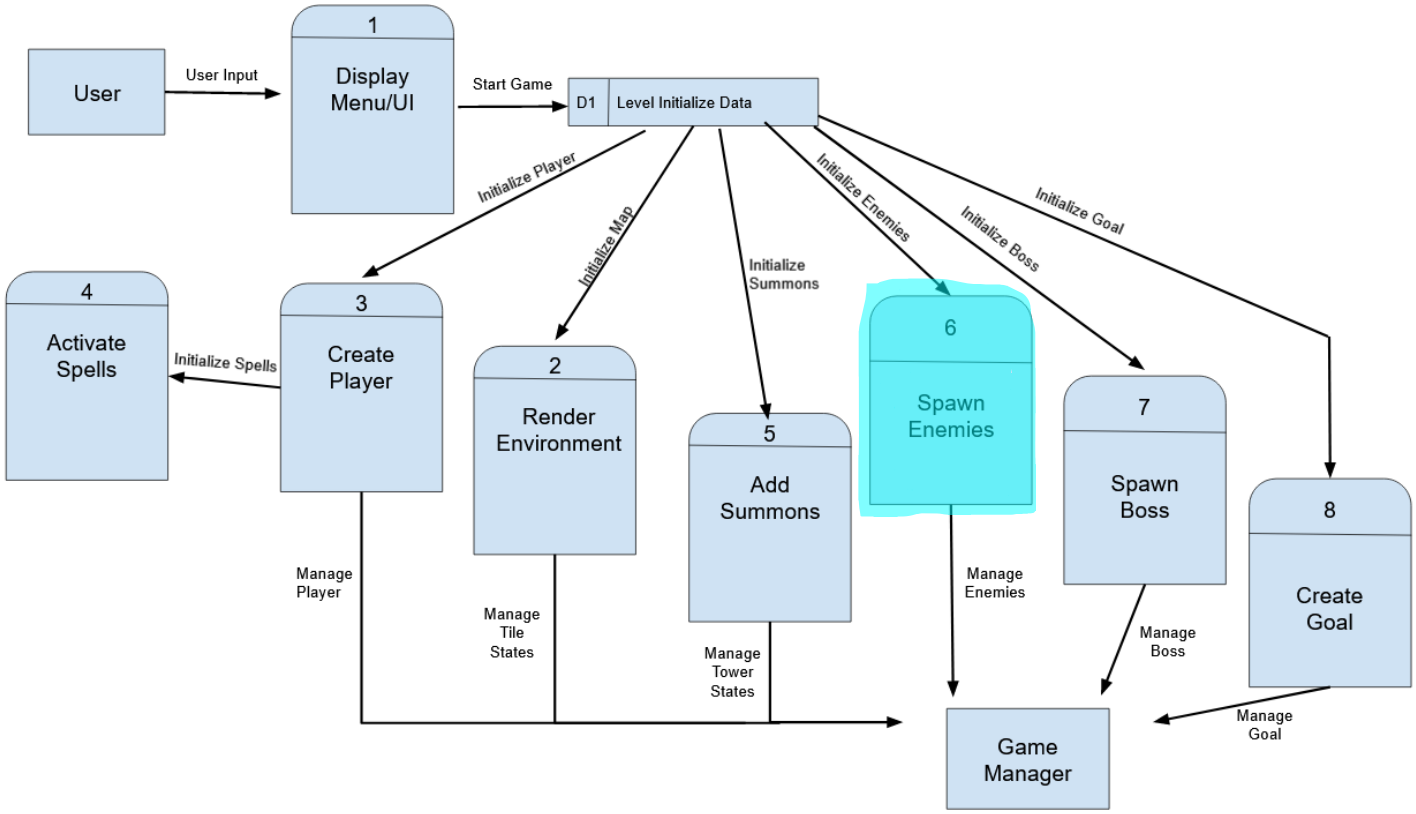
Post conditions: All enemies are terminated.
Priority: 2*
ID: IE1
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

3. Data Flow diagram(s) from Level 0 to process description for your feature _____14

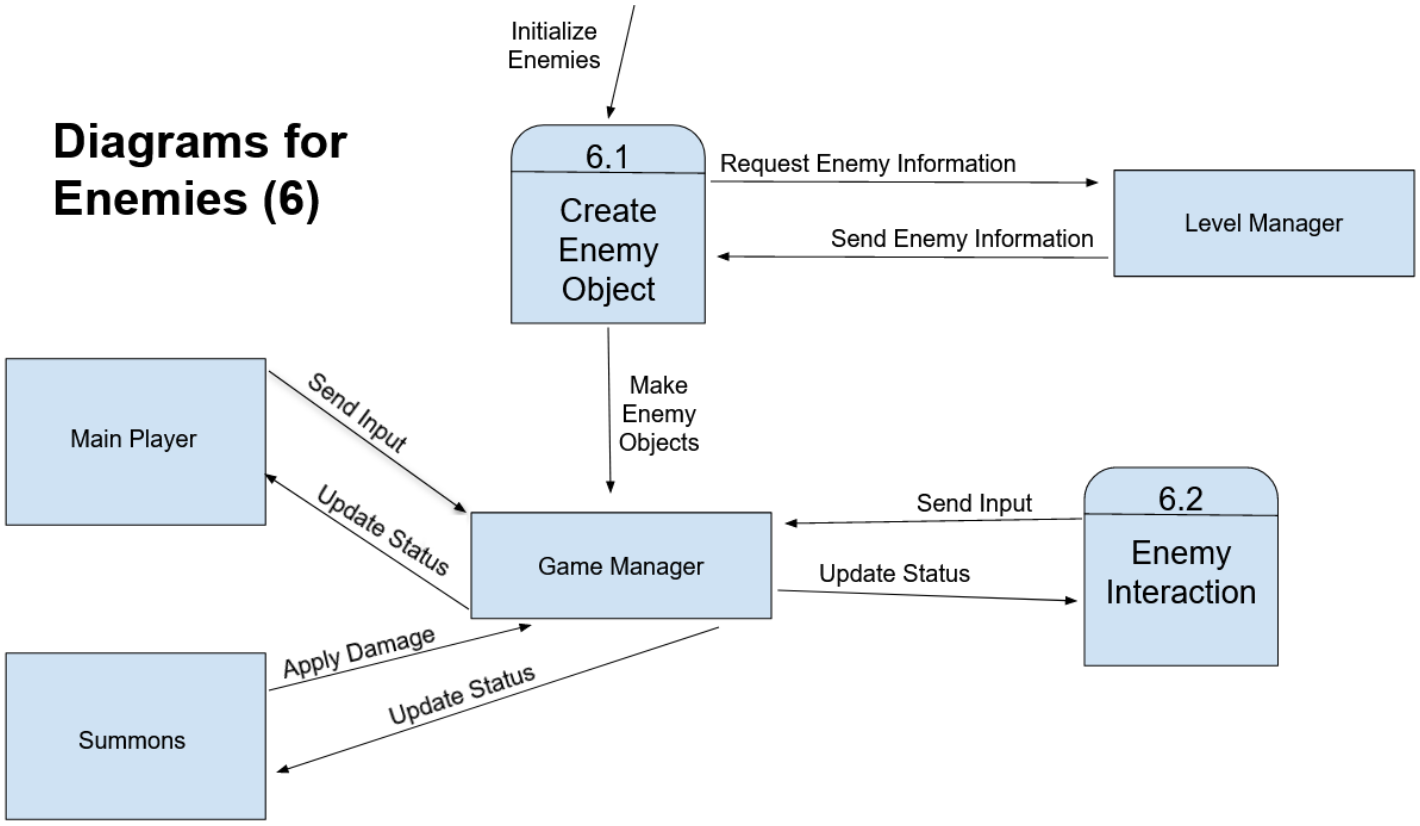
In the Data Flow Diagrams below, I will be covering the Enemy feature in depth. The two main components are creating enemies and managing enemy interactions.

Data Flow Diagrams

Diagram 0



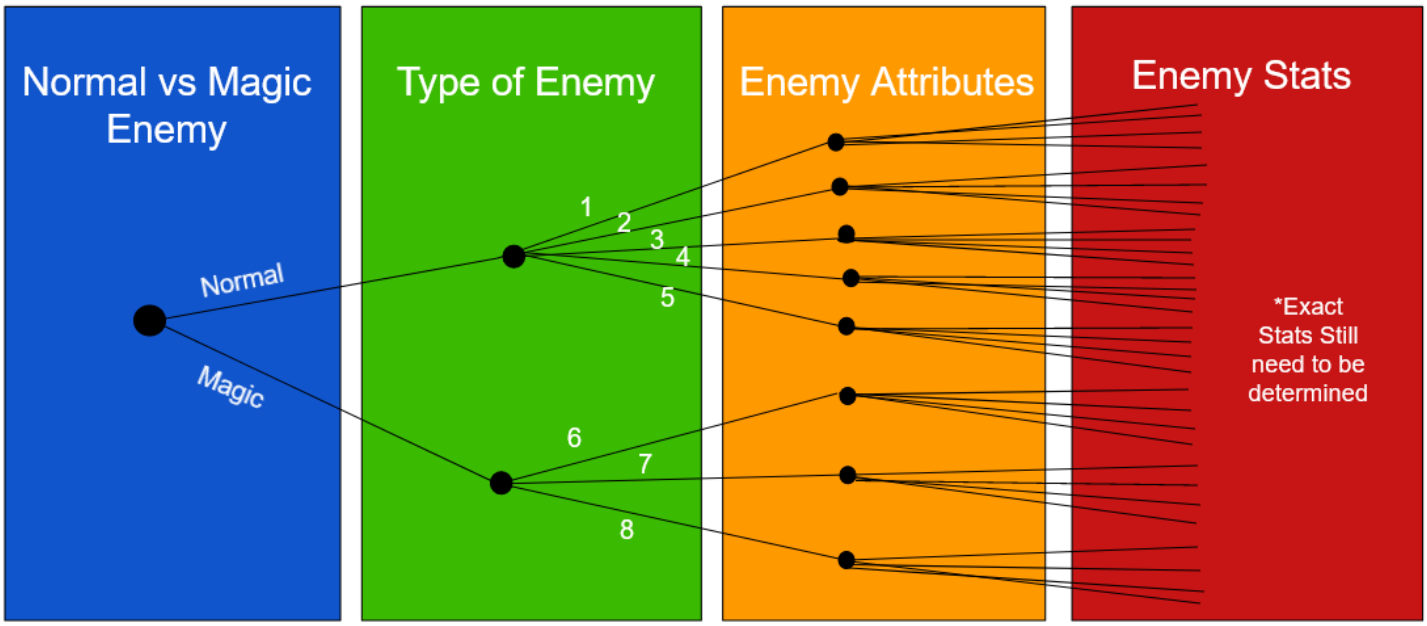
Diagrams for Enemies (6)



Process Descriptions

The process description for Process 6.1 (Create Enemy Object) is displayed below in a decision tree.

Process Description Decision Tree (Process 6.1 Create Enemy Object)



Type Key:

1 - Goblin Grunt
2 - Goblin Warrior
3 - Goblin Archer
4 - Goblin Assassin
5 - Goblin Giant
6 - Slime-caster
7- Goblin Alchemist
8 - Goblin Priest

Attribute Key:

A - Health
B - Damage
C - Speed
D - Ability

4. Acceptance Tests _____9

This feature will have a few different aspects which each need a unique acceptance test. The first part that will need to be tested thoroughly will be the enemy object creation. Next, we will need to have another acceptance test for the enemy's interactions with the main player and summons. Finally, we will also need to test the enemy's pathfinding to ensure it has found the correct path.

The following sections will describe each of these acceptance tests in depth.

Enemy Creation:

For creating the enemy objects, we need to ensure that the number of enemies, the type of enemy, and the stats for each enemy is correct. To do this we could implement an acceptance test that creates 1000 enemies and compares it to the number generated in the test scene. This would demonstrate whether our program is generating the correct number of enemies. Next to test all 7 types of enemies we could create 100 objects for each enemy type and compare them to the types generated in our test scene. When we test for the type of enemies, we could also check the stats of those enemies to certify they match the expected values. This acceptance test would provide us with enough information to determine if our program is able to create enemies successfully.

Enemy Interaction:

In this acceptance test we are going to check the enemy's interactions with the main player and summons. First, we will spawn 100 enemies in our test scene. Next, we will check the main player interaction through damaging the enemy characters and we will also need to have the enemies attack the player to show that their damage output is working. We will have to have different values like 1, 5, 10, 20 , 50, 100, 1000 for the damage output of the main player and compare it with the damage received from enemy checking that the values are the same. Then will repeat this step but instead have the enemy dealing damage to the player. Its also important to note that when the enemy unit or main player are dealt lethal damage that the correct event will happen. For the summons we can test the damage output the same way we did above for the player. This acceptance test should ensure our interactions are working correctly.

Pathfinding:

The acceptance test for our pathfinding algorithm will use pre-generated maps that we have the solution for. We can give our algorithm the empty test map file that will contain impassible objects, a start position, and a target position. Then our pathfinding algorithm will find the shortest path from the start position to the goal position and return the correct path. We can then compare this map with our solution map to see if the algorithm is operating correctly.

5. Timeline _____/10

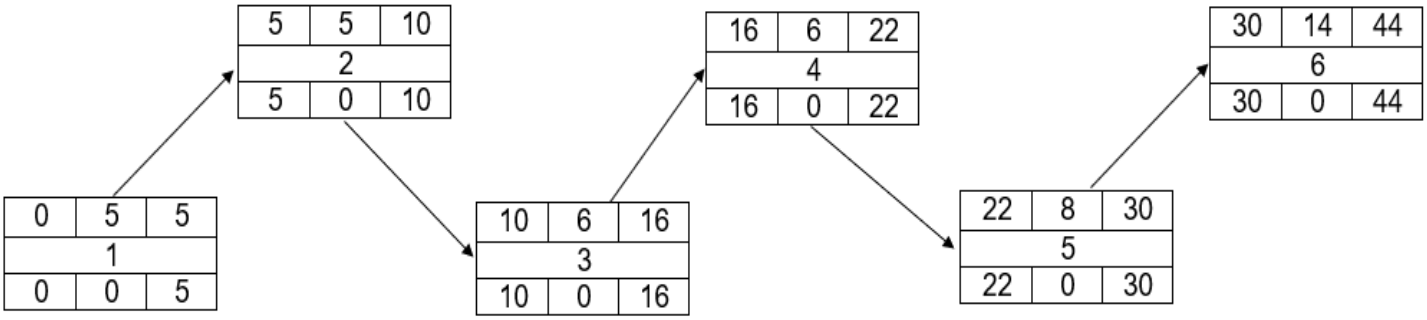
Work items

Task	Duration (Hours)	Predecessor Task(s)
1. Enemy Design	5	-

2. Enemy Characteristics	5	1
3. Enemy Movement/Pathfinding	6	2
4. Enemy Interaction	6	3
5. Enemy Targeting	8	4
6. Game Demo Mode	14	5

Pert diagram

David Bush:



Gantt timeline

David Bush:

Key:

Work Hours

Hours