Name: Ethan Corgatelli Mark ___/50

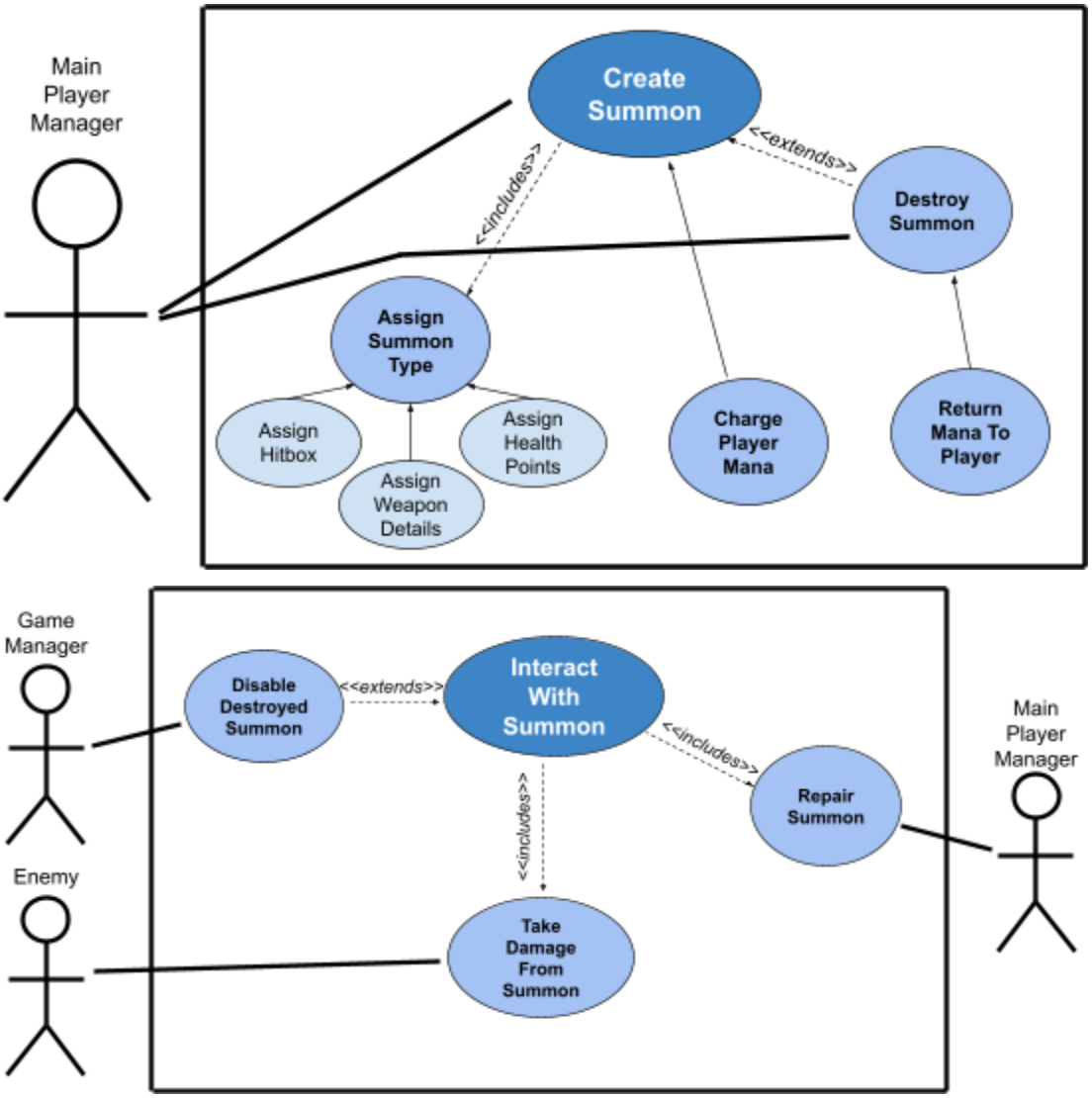
1. Brief introduction __/3

My champion feature for *Wonky Wizards* is the Summons a.k.a. towers/automatic defenses. Before each wave of enemies, there will be a preparation phase in which the player can spend their mana on summons. A list of basic summon types: Barrier, Ghost Pepper, Enchanted Swords, Astral Crossbow, Portal, Empowering Crystal, Shrubbery from NI, Sven, Mysterious Futuristic Wind Manipulation Device, and finally Big Chuck. These summon types all have their own set of statistics, including: Cost, Projectile Travel/Damage/Firing Speed, HP, Effect.

I will also be creating (at least finding and/or designing) all of the sounds in the game, including the sound effects and music.

2. Use case diagram with scenario __/14

Use Case Diagrams



Scenarios

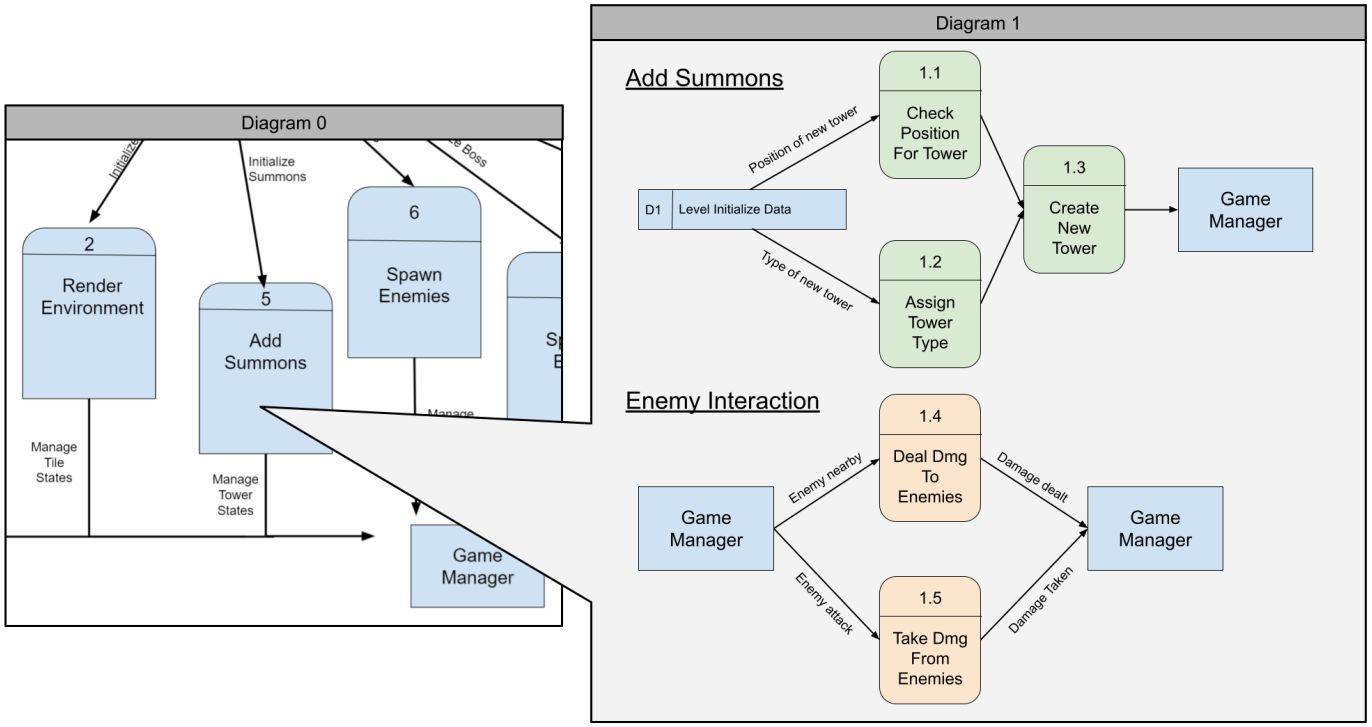
Name: Create Summon
Summary: During the setup phase, the player spends mana to buy and place summons.
Actors: Main Player Manager script (*i.e.: the player of the game*).
Preconditions: Game is in the setup phase.
Basic sequence:
Step 1: The player uses a menu to pick from the list of available summons based on the amount of mana they have.
Step 2: The player chooses where in the level to place the summon by walking to the tile where they would like to place it.
Step 3: The appropriate amount of mana is taken from the player.
Step 4: The summon is placed into the world by the player.
Step 5: The summon's type is assigned: its hitbox, weapon details, and healthpoints are all assigned based on what kind of summon is being placed.
Exceptions:
Step 1: If the player tries to buy a tower that they do not have enough mana for, they will be given an error message and told to try again.
Step 4: If the player tries to place a summon on an invalid tile (e.g. on top of another tower or out of bounds) this will not be allowed, so that summons cannot overlap.
Post conditions: The player now has one more summon to help them defend their territory.
Priority: 1*
ID: EC1

Name: Interact With Summon
Summary: Possibilities of an interaction with a Summon include: giving damage & taking damage (actions of the Enemy), and repairing a summon (action of the Main Player).
Actors: Game Manager, Enemy, Main Player Manager.
Preconditions: Valid summon is in the world
Basic sequence:
Step 1: If an enemy or the player is within the active radius of a summon, they can interact with that summon.
Step 2: Enemy takes its damage from summon.
Step 3: Player repairs summon (if applicable to use).
Exceptions:
Step 2: If the enemies heal goes to/below 0 after the damage is dealt, it should die and be removed.
Post conditions: Interaction has been made (i.e.: summon repaired/gives damage/takes damage).
Priority: 2*
ID: EC2

*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

Data Flow Diagrams



Process Descriptions

- Add Summons — two pieces of information are retrieved from the level initialize data:
  - 1.1 Check Position For Tower (takes position for the new tower being placed) — Checks that the given position is valid (e.g. is not a wall in the level layer, or already has a tower on it).
  - 1.2 Assign Tower Type (takes the type of requested tower) — assigns the type for the new tower being placed.
  - 1.3 Create New Tower (the information in 1.1 & 1.2 is combined) — Sends the new tower to be placed in the world to the Game Manager.
- Enemy Interaction:
  - 1.4 Deal Damage To Enemies (takes nearby enemy position) — deals damage to the enemy by sending the information to the Game Manager.
  - 1.5Take Damage From Enemies (stretch goal: takes enemy attack) — applies enemy attack's damage to the tower.

4. Acceptance Tests __/9

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

Summons Test #1:

Input scenario:

On a blank default map, cover anywhere from a single space, all the way up to every space with a summon of a random type, always being sure to leave at least a single wide path from the enemy spawn point for enemies to get to the goal tower (located in the center of the map).

Run waves of many enemies (as if a real game is being run - but with no character player), allowing for the performance of the towers to be tested. During the duration of the test Unity's Profiler can be used to monitor the performance of the summons. Start with a single summon, and run tests all the way up to every available summon position, recording the performance of each run.

Expected output:

If all goes well, the performance in the later tests (with more summons) should not be significantly worse. If the performance is heavily impacted from having more towers, this may indicate that I need to rework something about how the summons are written, or that I may need to optimize something.

Task

Duration (PWks)

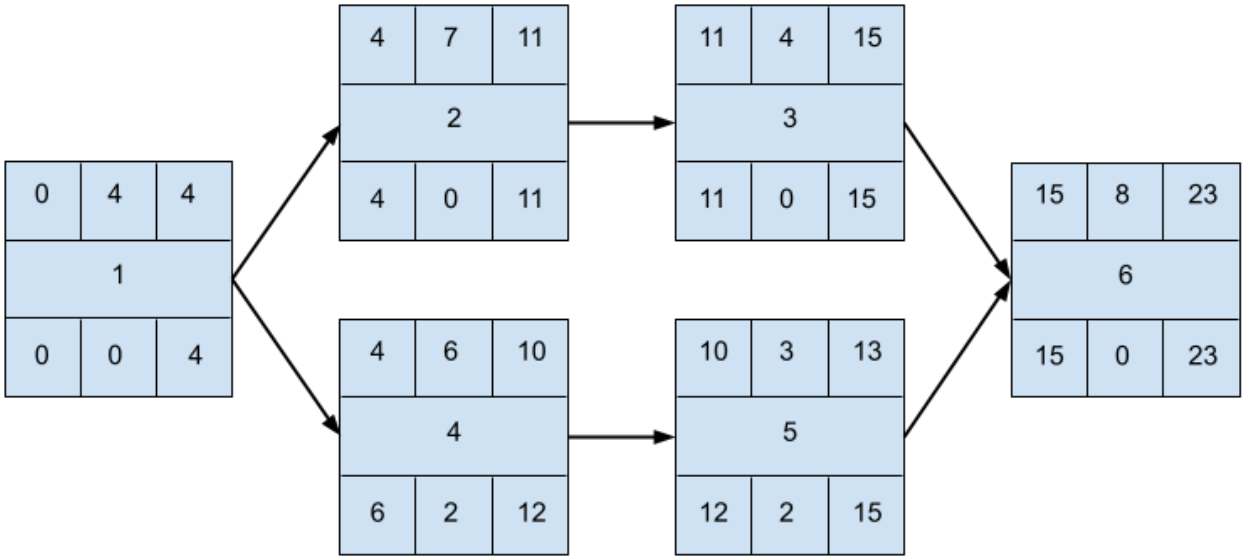
Predecessor Task(s)

5. Timeline __/10

Work items

#	Task	Estimated Duration (Hours)	Prerequisite Tasks(s)
1.	Create Summon Masterclass	4	N/A
2.	Create Subclass Summon Types	7	1
3.	Types Testing/Balancing	4	2
4.	Create Game Music	6	1
5.	Create Game Sound FX	3	4
6.	Mixing Sound/Bug Fixing	8	1, 4

Pert diagram



Gantt timeline