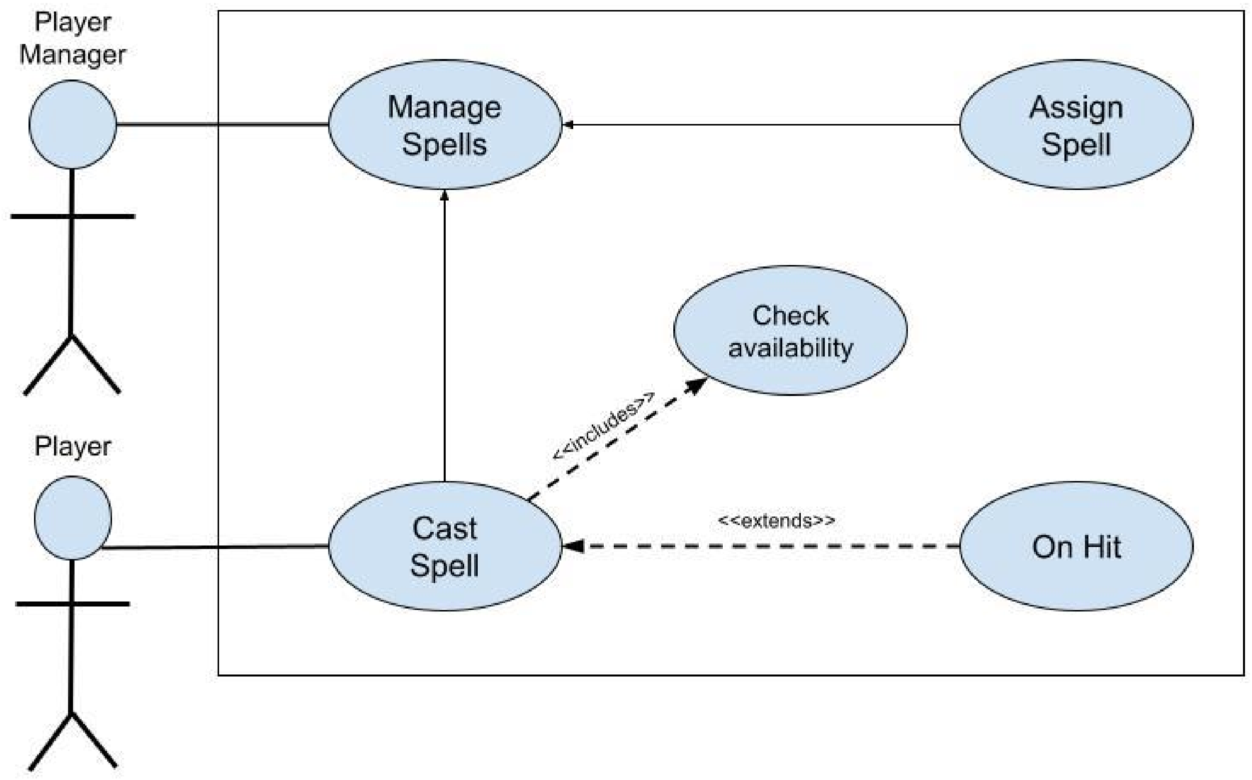
Zach Heimbigner
Champion Feature: Spells
Team 7: Bloonigans
9/23/2021

Introduction:

My feature is creating spells for the player to use in order to help fight off the horde and defend the goal, as well as keeping the player alive. Each spell will have to be purchased from the store in order to use, then the player will be able to use said spell throughout the level and subsequent levels.

I am also responsible for creating the game manager for this game. I have included the time needed for this feature as well in my timeline and tests.

Use case Diagram with Scenarios:



Scenario:

Name: Spell Feature

Summary: The player uses the UI to select a spell they wish to use, and then the player can cast that spell as the current spell or select another

Preconditions: Spell class is initialized.

Actors: Player, PlayerManager

Basic sequence:

1. Player Selects Spell they wish to use, Player manager makes the request
2. Assign Spell is called and the desired spell class is called
3. The instance is passed back to the spell manager and set as current spell
4. Player casts the desired spell
5. Check if spell is on cooldown or not
6. Cast spell

Exceptions:

Step 1: [cast spell] is pushed and the spell is not available: deny selection .

Step 2: A button other than [select/cast spell] is pressed: ignore input.

Step 3: Player selects another spell on cooldown: select but do not cast.

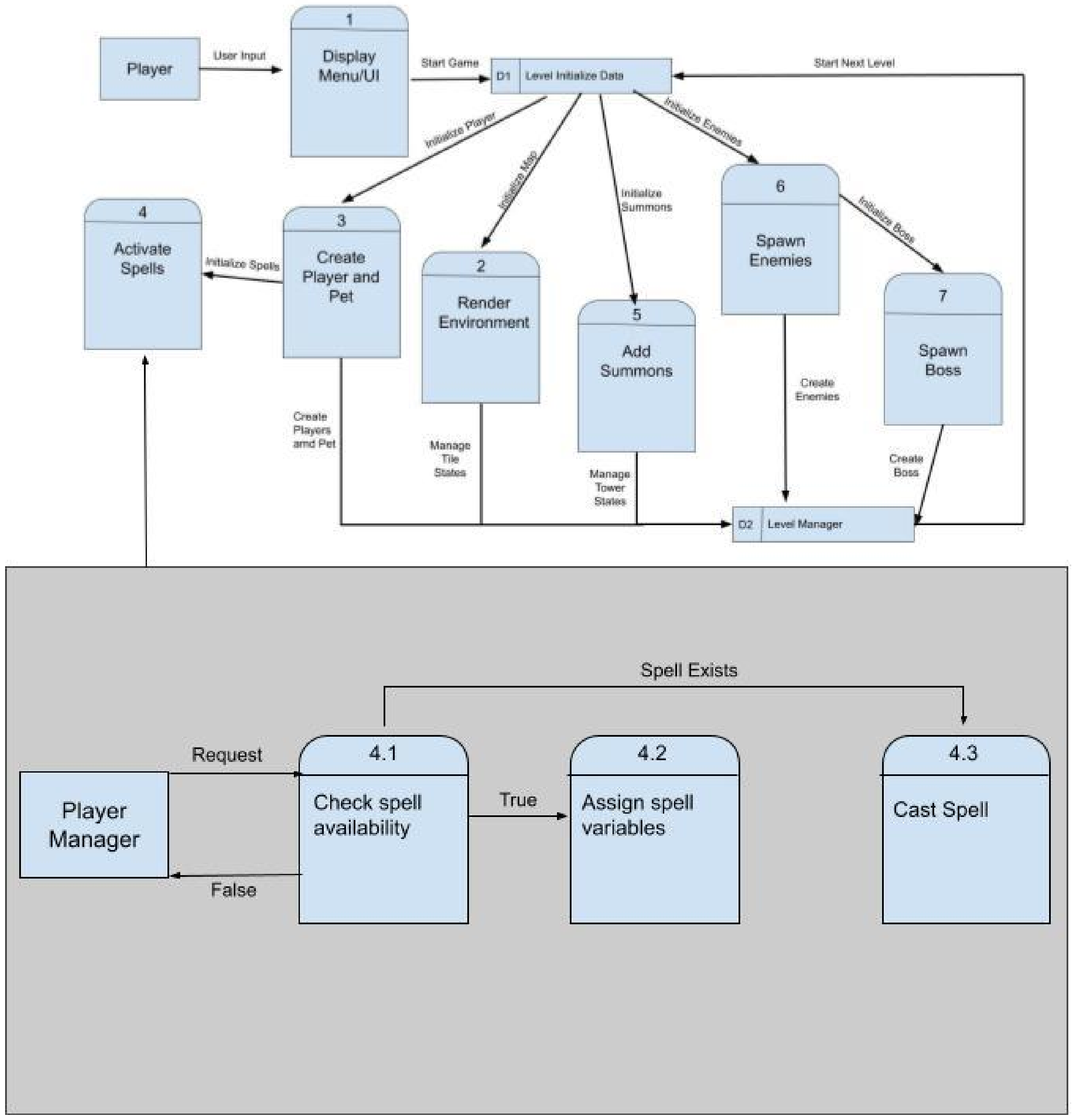
Step 4: Player casts spell in front of wall: spell hits wall and is deleted.

Post conditions: Spell selected and ready to cast.

Priority: 2*

ID: Z01

Data Flow Diagrams:



Process Descriptions:

The processes are written is pseudo code below:

- *Check Spell Availability:*

```
WHILE GameRunning:
    Await select spell
END WHILE

Await select spell:
    Take in requested spell
    Is Current Spell:
        Yes:
            Cast Spell
        No:
            Check queue to see if spell is available:
                Not in queue:
                    Deny request
                In queue:
                    Accept request
                    Call Assign spell
END await spell
```

- *Assign Spell Variables:*

```
Take in Spell Request
Remove current spell instance
Locate class of spell //do not need to check if it already is called, handled above^
Create instance of that class of spell
Return instance as current spell
```

- *Cast Spell:*

```
Take in instance of current spell
Create GameObject from prefab
Add force to Gameobject in Direction of the mouse
Put spell on cooldown
```

Acceptance Tests:

Test #1:
Cast spell n amount of times:

The Game will behave as such:

1. The Game will create a game object
2. The GameObject will travel in the direction of the cursor
3. The Game Object will disappear when it collides with something
4. If it collides with an enemy it will damage the enemy
5. If it collides with a wall then it will just disappear.
6. The spell will go on cooldown, it cannot be cast until it is off cool Down
7. Repeat steps 1-6 for n amount of casts

The test fails if:

Any of the steps are not met when casting, or yield a separate result.

Test #2
Rotate spells and cast

The Game will behave as such:

1. The player will select another spell
2. The game will load the spell
3. The player casts the spell (see test #1)
4. The player rotates spells (if others are available)
5. The player casts the second spell

The test fails if:

- Any of the steps are not met when casting, or yield a separate result.
- When switching spells the cooldowns get reset
- If the player can cast 2 spells at once
- The player can rotate between spells they have not bought

Test #3:
The spell acts as intended when colliding with enemies

The Test Fails if:

- The behavior is not the assigned behavior.
- The spell enacts the behavior when not hitting an enemy.

Test #4: GameManager

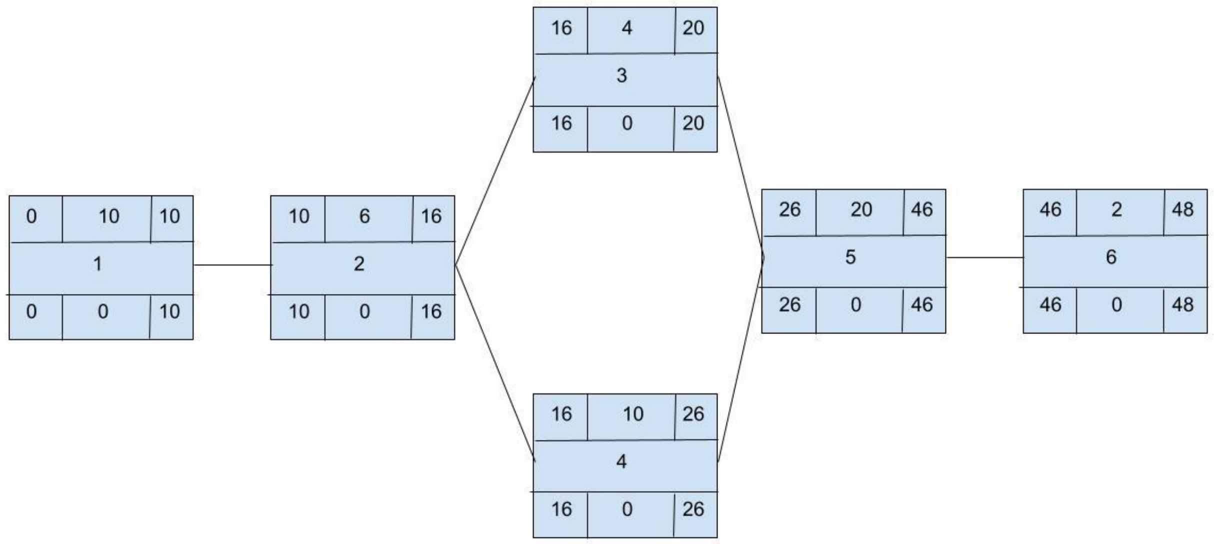
The game manager will control the flow of the game and initiate the sequence of events needed for levels and overall gameplay. Testing this feature will be about looking at its behavior and modifying to meet the needs of my team members. Each team member has a feature that goes into this game and I will need to meet with each member of my team to choreograph how this game should flow. Tests will include:

- State change tests to ensure the game flows properly
- State tests to ensure each state does the appropriate function
- Efficiency tests, to ensure the game manager can handle all it manages in an Efficient and clean manner.

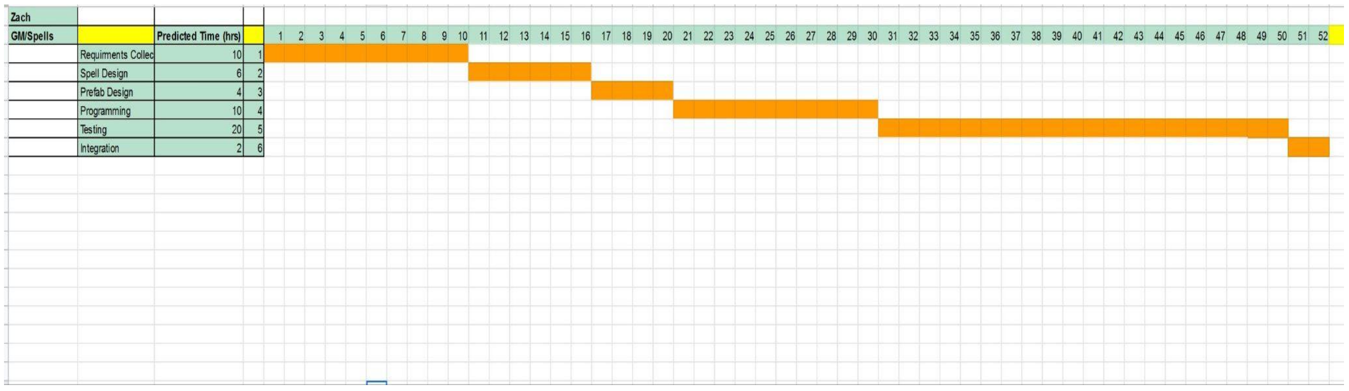
Timeline:

Task	Duration(PWks)	Predecessor Task(s)
1 Requirements Collection	10	-
2 Spell Design	6	1
3 Prefab Design	4	2
4 Programming	10	2
5 Testing	20	4
6 Integration	2	4,5

Pert Diagram:



Gantt Chart:



Zach			
GM/Spells		Predicted Time (hrs)	
	Requiments Collec	10	1
	Spell Design	6	2
	Prefab Design	4	3
	Programming	10	4
	Testing	20	5
	Integration	2	6