Name_________Joshua Tan___________ Mark ______________________/50

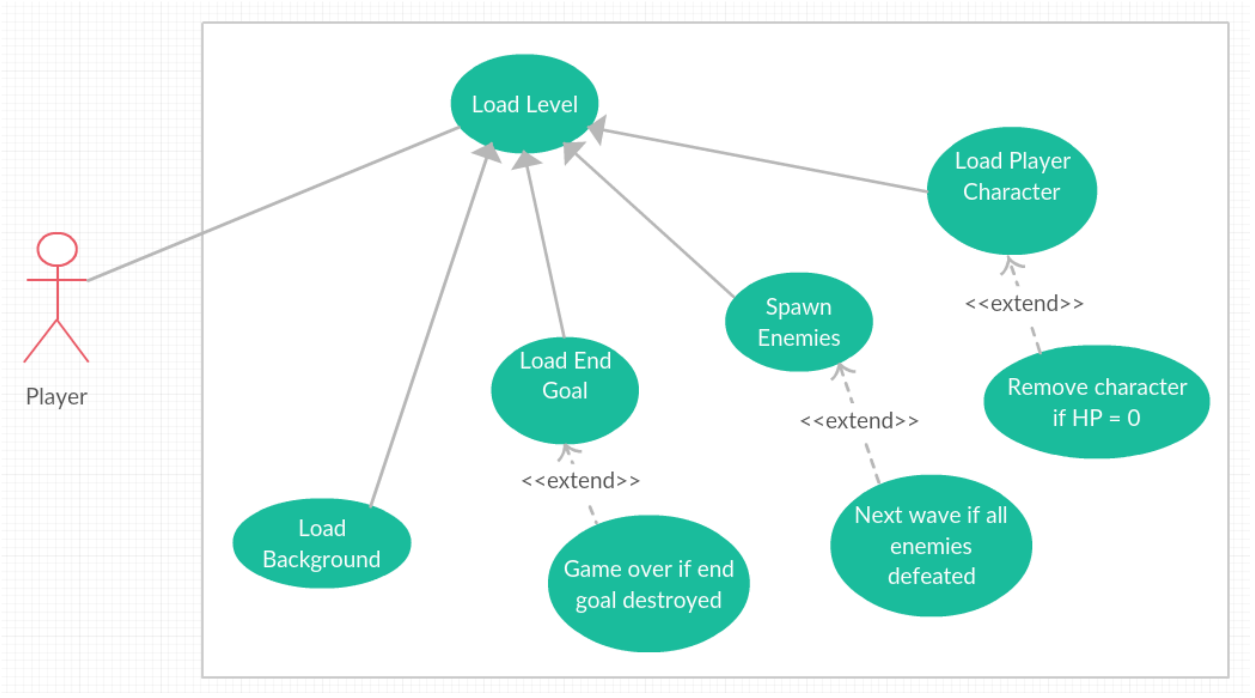
[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

1. Brief introduction __/3

My feature for the Debugger game will be the level. Enemies will spawn from the right side of the level and move progressively towards the left. The player-controlled character will spawn at the same time the level is loaded. Since our game will be a tower defense game, the level will be set up so that when enemies successfully reach a certain portion of the field the player will lose points. The player-controlled character can move freely throughout the boundary of the level. The player will also be able to build defensive structures in the level. The level doesn't change through the different waves the player may encounter until the game is over.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

Name: Load Background

Summary: Unity engine loads the background scene.

Actors: Player.

Preconditions: Player must start the game.

Basic sequence:

Step 1: Receive scene information to be loaded.

Step 2: Continue to display scene until player exits the game.

Exceptions:

Step 1: Game is paused: Display pause menu.

Step 2: Game is over (player loses): Display game over screen.

Post conditions: Visual representation of the playing field is displayed.

Priority: 2

ID: C01

Name: Load End Goal

Summary: Unity engine loads the end goal(s).

Actors: Player.

Preconditions: Player must start the game.

Basic sequence:

Step 1: Receive object information to be loaded.

Step 2: Load and display the object at the designated point the level.

Step 3: Remove the end goal object if it is destroyed

Step 4: Display game over screen if all end goals are destroyed.

Exceptions:

Step 1: End Goal object couldn't be loaded. Display Error message.

Post conditions: Visual representation of the end goal(s) and their status (alive or destroyed) is displayed.

Priority: 1

ID: C02

Name: Spawn Enemies

Summary: Unity engine loads the enemy objects.

Actors: Player.

Preconditions: Player must start the game, not "Game Over".

Basic sequence:

Step 1: Receive enemy object information to be loaded.

Step 2: Spawn enemies according to the predetermined amount according to the wave the player is currently on.

Step 3: If all enemies are defeated for the current wave, proceed to the next level.

Exceptions:

Step 1: Enemy object couldn't be loaded. Display error message.

Post conditions: Visual representation of the enemies are displayed.

Priority: 1

ID: C03

Name: Load Player Character

Summary: Unity engine loads the player character object.

Actors: Player.

Preconditions: Player must start the game, not "Game Over".

Basic sequence:

Step 1: Receive player character object information to be loaded.

Step 2: Check for collisions along the screen border to ensure player character stays within the playing field.

Step 3: If player is defeated, remove player object and display game over screen.

Exceptions:

Step 1: Player character object couldn't be loaded. Display error message.

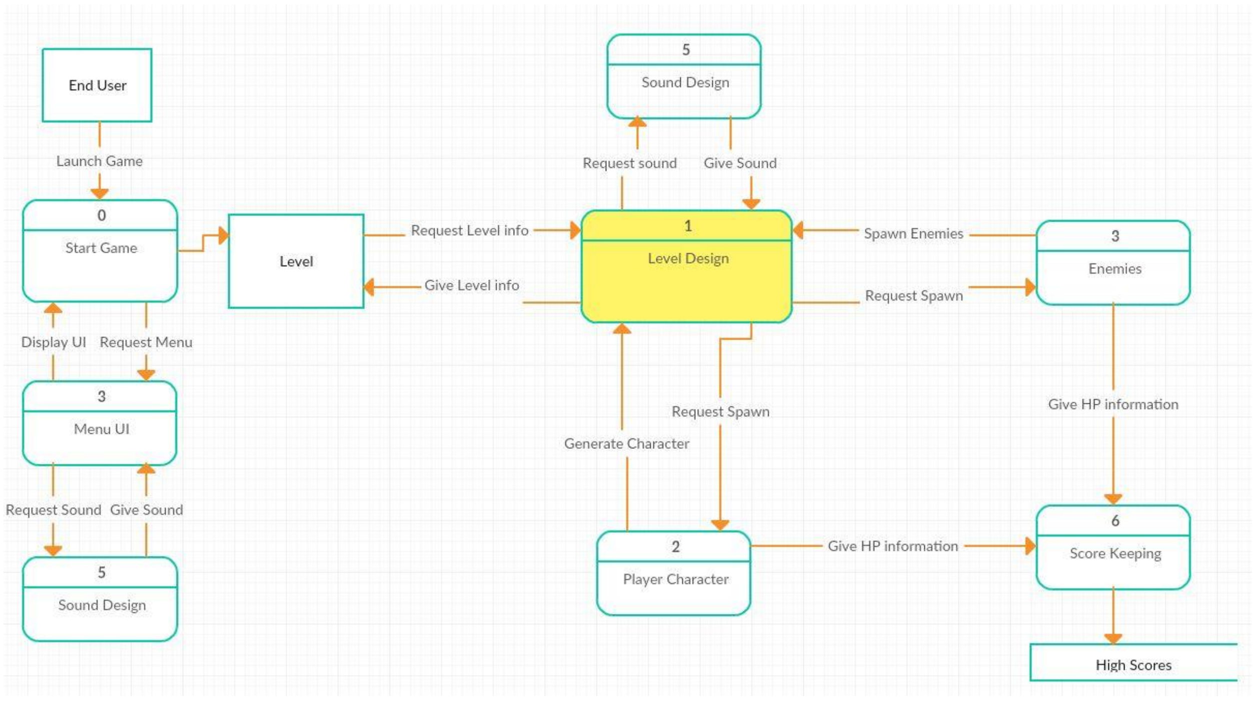
Post conditions: Visual representation of the player character is displayed.

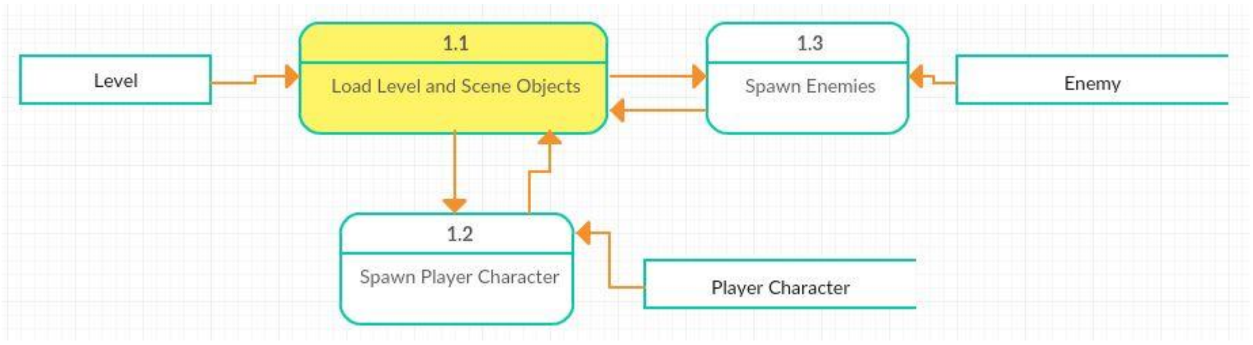
Priority: 1

ID: C04

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Data Flow Diagrams





Process Descriptions

Load Level and Scene Objects:

```
WHILE game is not over OR no menu UI are open
    Display scene design
    Display scene objects (end goal)
    Display Player Character
    Display Enemies
    IF Player Character dies OR end goal is destroyed
      Remove Player Character
      Display Game Over UI
    ENDIF
    IF Enemies for current wave are defeated
      Display “Next Wave” text
END WHILE
```

4. Acceptance Tests _______9

Test Load Scene Feature

Scene should only be loaded once per game (from new game to game over). Game will be restarted 10 times to ensure only one instance of the scene is loaded at a time.

Test End Goal Feature

When the end goal is damaged or destroyed, it’s state should stay the same until the player reaches “Game Over”. This can be tested by running the game with 10 waves of enemies that will slowly damage the end goal, eventually leading to game over by wave 10 if the test is successful.

Test Collison Feature

Generate 50 enemies and 10 player-controlled structures.

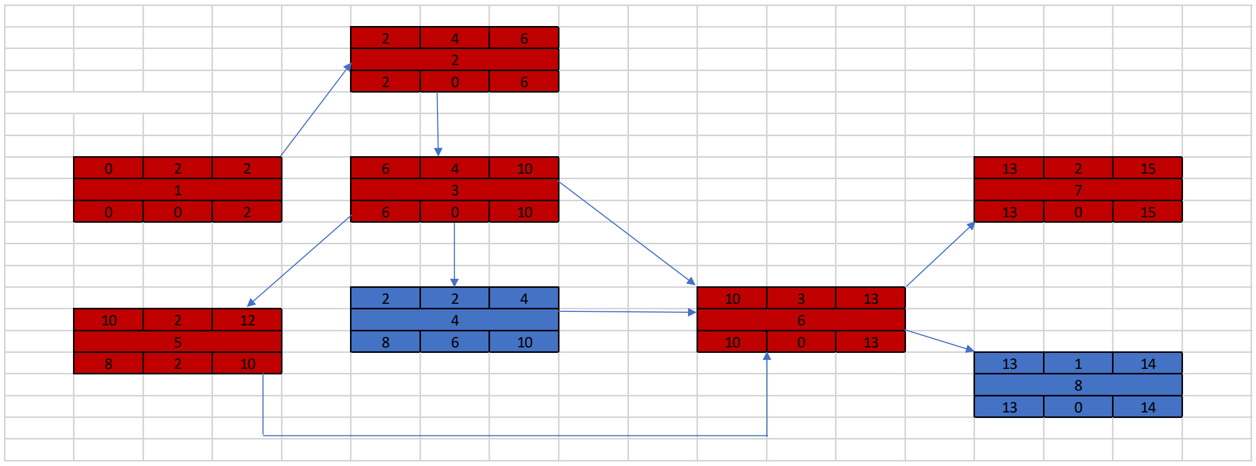
Enemies should not be able to pass through player-controlled structures. Collisions between the player’s character and the enemy should result in the player taking damage. Collisions between the enemy and the end goal should cause the end goal to take damage. No objects may leave the boundary of the playing field (the entire screen). No character or structure should take damage if there are no collisions.

5. Timeline ________/10

Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Define level size and ratios for objects	2	-
2. Generate visuals (scene, structures)	4	1
3. Program physics colliders (scene, structures)	4	2
4. Spawn player character at game start	2	3
5. Spawn enemies as per wave	3	3
6. Testing and Debugging	3	3,4,5
7. Documentation	2	6
8. Release Build	1	6

Pert diagram



Gantt timeline

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	🟥	🟥																
2			🟥	🟥	🟥	🟥												
3							🟥	🟥	🟥	🟥								
4							🟦	🟦	🟦	🟦								
5											🟥	🟥	🟥					
6														🟥	🟥	🟥		
7																	🟥	🟥
8																	🟦	