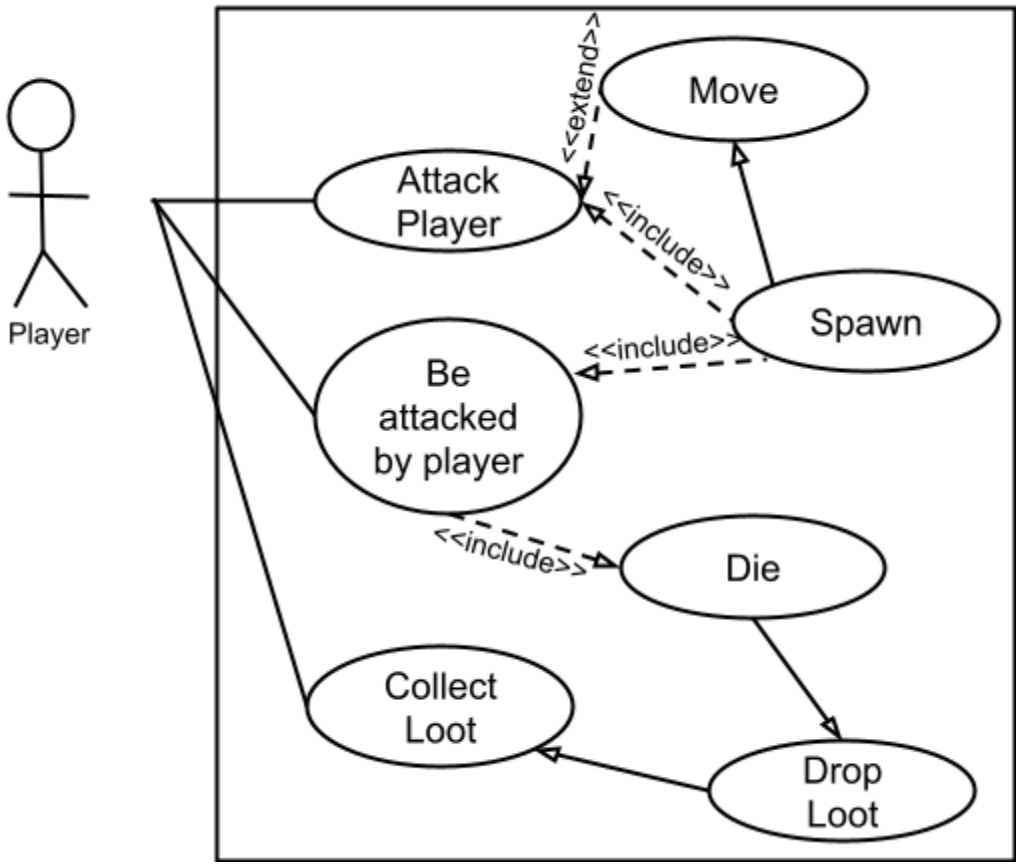
Name Kynan Murphy Mark _____/50

1. Brief introduction __/3

I will be adding enemies into the game. The enemies will be able to spawn into a map when the map is generated, will be able to move around freely in the map, and will be able to attack the player.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

[You will need a scenario for each use case]

Name: Attack Player

Summary: The player is attacked by enemies in the game

Actors: Player

Preconditions: The enemy has been spawned in

Basic sequence:

Step 1: Wait until attack cooldown is at 0

Step 2: Identify location of player relative to enemy

Step 3: Shoot at the player

Step 4: Reduce the player's health

Post conditions: Attack cooldown resets

Priority: 2*

ID: C01

Name: Be attacked by player

Summary: The player attacks the enemy

Actors: Player

Preconditions: The enemy has been spawned in

Basic sequence:

Step 1: Calculate damage radius of player's attack

Step 2: Lose Health

Exceptions:

Step 1: The enemy isn't in the attack radius: Don't lose health

Step 2: The enemy's health reaches 0: Die

Post conditions: Enemy Loses health

Priority: 2*

ID: C02

Name: Collect Loot

Summary: The player collects loot dropped by the enemy

Actors: Player

Preconditions: The enemy has dropped loot

Basic sequence:

Step 1: Check if loot is in player's collect radius

Step 2: Increase player's count of said loot

Step 3: Remove instance of self

Exceptions:

Step 1: Player is too far from loot: Skip following steps

Post conditions: Player gains loot

Priority: 3*

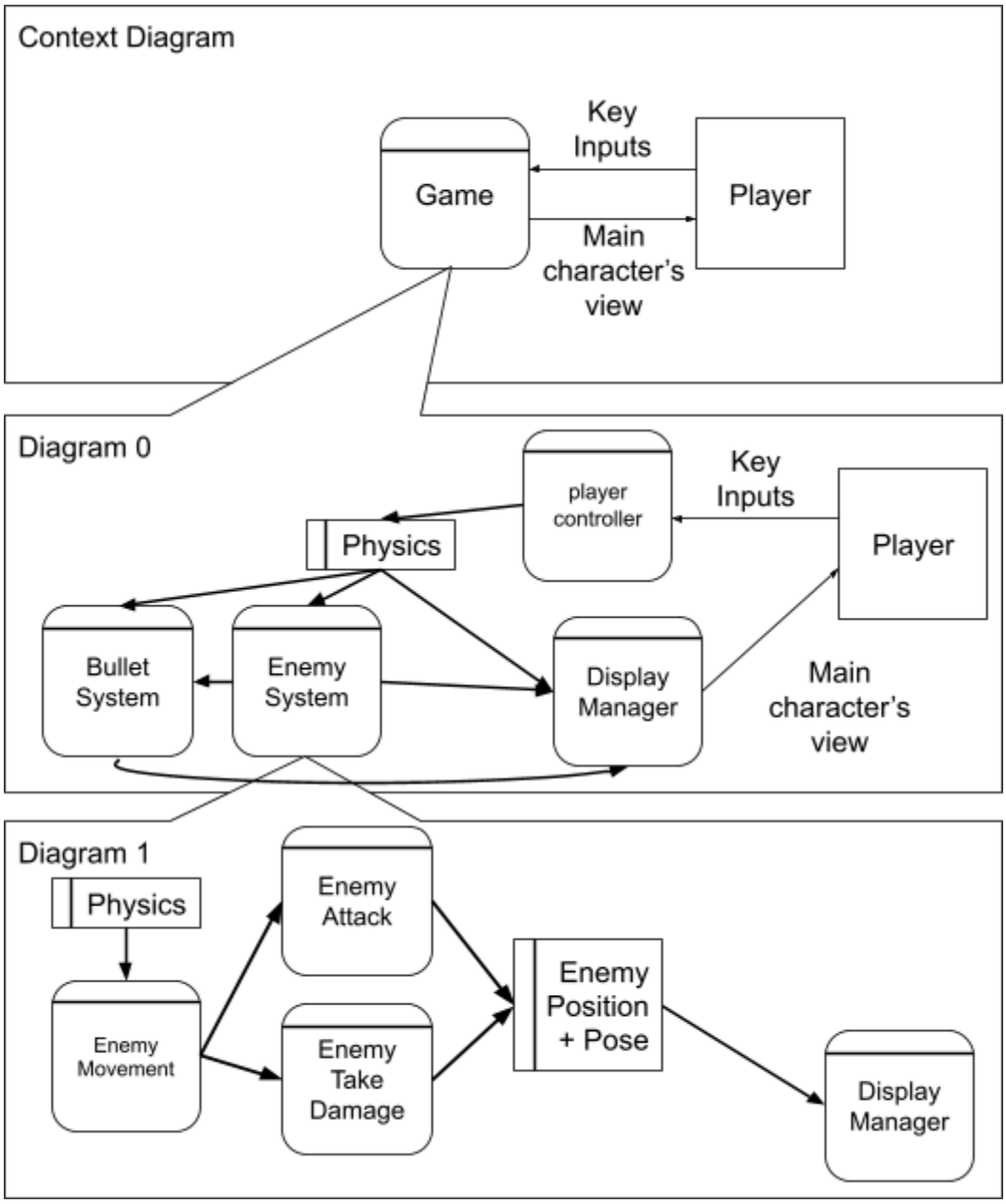
ID: C03

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

[Get the Level 0 from your team. Highlight the path to your feature]

Example:

Data Flow Diagrams



Process Descriptions

Enemy Movement*:

WHILE enemy knows where player is

Find Shortest path to player

View map as sections of connected pieces, where all obstacles are excluded, and the adjacent pieces that the enemy can travel to and from are linked to each other. The piece the player is on is given the value 0, and recursively each map piece is given the value of its smallest neighbor +1. This process stops when there are changes made in a step.

If the spot the enemy is on has no value, there is no path to the player.

If the spot has a value, the enemy scans all adjacent pieces and identifies the one with the smallest value. The enemy then heads in a direct line towards that piece.

Move to player

WHILE enemy doesn't know where player is

Search

Randomly move and turn

If player in site radius and in focal cone, send an invisible bullet towards the player's head at a rate of about 5% of the distance per frame.

If the bullet reaches the coordinates it was sent to, the enemy sees the player. If the bullet collides with an obstacle, the enemy doesn't see the player.

If the enemy sees the player, set the enemy's known location of player to the current player location for as long as the player is in the enemy's focal cone and at least n seconds, before the enemy loses the player again, and sets the last known location of the player as its destination.

If the enemy lost site of the player and successfully reached the last known site without finding the player again, reset the known location to undefined and start wandering again.

Enemy Attack*:

WHILE enemy is in attack range of player, and player is in view

Attack Player

Enemy Take Damage*:

IF enemy is hit by player attack

Take damage

Automatically find the player and set last known coordinates to player location

4. Acceptance Tests _____9

Path Finding

Randomly generate maps where there are disconnected and connected pieces, and run the path finding function, outputting to a file the path to the player, as well as an adjacency matrix of the maps.

The expected outputs are:

- A path showing the shortest from the enemy to the player
- An empty list when the enemy is on a disconnected piece of map

Player spotting

Randomly generate a terrain with obstacles and put the player and enemy at random locations. Shoot the invisible bullet from the enemy to player, and compare the results with a more slowly shot bullet to test the accuracy of the method.

The expected outputs are:

0, 0, when there is an obstacle

1, 1, when there is no obstacle, or when both methods fail to detect an obstacle

1, 0, when there is an obstacle, but the less accurate method can't find the obstacle.

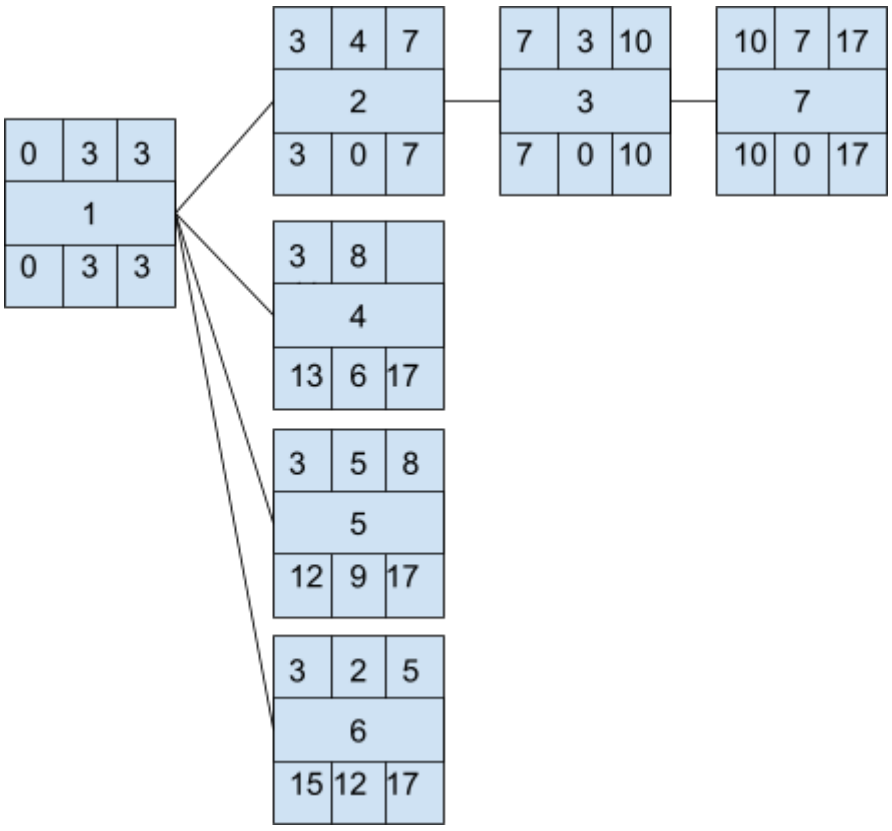
This test will not be run with the intention of completely eliminating errors, but with the intention of minimizing errors without decreasing efficiency too much.

5. Timeline _____/10

Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Spawn / Stats	3	-
2. Enemy Weapons / Bullets	4	1
3. Enemy Attack	3	2
4. Enemy Pathfinding	8	1
5. Enemy Searching	5	1
6. Damage / Death	2	1
7. Attack Logic	7	3

Pert diagram



Gantt timeline