Name: Damon E Schafer Mark _____________________/50

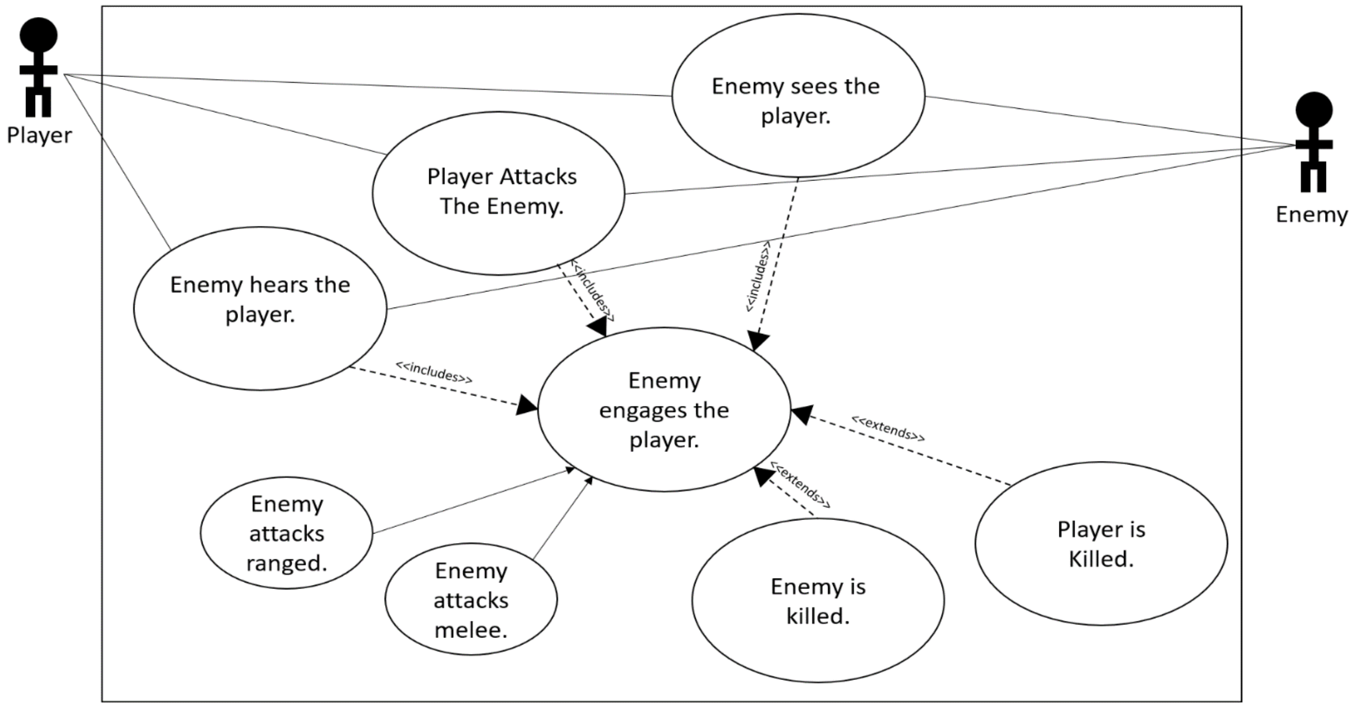
1. Brief introduction __/3

The feature that I have chosen is the implementation of enemies and enemy AI. This includes creating a system that facilitates the creation of different kinds of enemies that the player can face. Each enemy will have unique traits and behavior such as different AI movement, aggression, speed, health, and attack types. Enemies can both harm the player and be harmed by the player. The enemies will also collide with the map and other objects present in the final game.

Each enemy will be represented by some textures and sprites that will be drawn into the current scene and respond to the environment.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

Name: Enemy hears the player

Summary: The player gets too close to the enemy, and the enemy becomes aware of the player's existence and enters an attack state.

Actors: Player, Enemy

Preconditions: Player is alive, the Enemy has been created and initialized, the enemy has not noticed the player yet.

Basic sequence:

> **Step 1:** Player gets within a set range of the player. This range will depend on each enemy, but it will be very close (like 5 tiles).
>
> **Step 2:** Enemy recognizes that the player is too close.
>
> **Step 3:** Enemy enters an attack/pursue state.
>
> **Step 4:** Enemy begins to attack the player, and/or get closer to the player to continue attacking while the player is still near and the enemy is still alive.

Exceptions:

> **Step 1,2,3,4:** Another game entity gets the enemy's attention before the player: the player has a higher priority and is pursued, and the other entity is ignored.
>
> **Step 3,4:** The player gets far enough away from the enemy for it to stop attacking/pursuing: the enemy enters its passive state.
>
> **Step 4:** The player gets killed when the enemy attacks it: the game enters its game over state.

Post conditions: Enemy attempts to pursue and attack the player.
Priority: 1
ID: E01

Name: Player attacks the enemy.
Summary: The player attacks the enemy, the enemy attacks the player in return.
Actors: Player, Enemy
Preconditions: Player is alive, Enemy is alive, the Enemy has been created and initialized.
Basic sequence:

> **Step 1:** Player attacks the enemy in a ranged or melee fashion.
>
> **Step 2:** If the player has not already been noticed, the enemy notices the player.
>
> **Step 3:** Enemy enters an attack/pursue state if not already in one.
>
> **Step 4:** Enemy begins to attack the player, and/or get closer to the player to continue attacking while the player is still near and the enemy is still alive.

Exceptions:

> **Step 1,2,3,4:** The enemy is attacking/pursuing another entity when the player begins to attack it: The enemy ignores the other entity and begins to attack the player.
>
> **Step 3,4:** The player gets far enough away from the enemy for it to stop attacking/pursuing: the enemy enters its passive state.
>
> **Step 4:** The player gets killed when the enemy attacks it: the game enters its game over state.

Post conditions: Enemy attempts to pursue and attack the player.
Priority: 1
ID: E02

Name: Enemy sees the player.

Summary: The player enters the line of sight of the enemy and is within the line of sight's max range.
Actors: Player, Enemy
Preconditions: Player is alive, Enemy is alive, the Enemy has been created and initialized, the enemy has not noticed the player
Basic sequence:

> **Step 1:** The player enters the line of sight of the enemy.
> **Step 2:** The player is within the max range of this line of sight.
> **Step 3:** The enemy notices the player and enters an attack/pursue state.
> **Step 4:** Enemy begins to attack the player, and/or get closer to the player to continue attacking while the player is in line of sight, and the enemy is alive.

Exceptions:

> **Step 1:** Something obscures line of sight while the enemy is seeing the player: the enemy enters a passive state.
> **Step 3, 4:** The enemy gets attacked by some other entity while pursuing the player: the enemy ignores this entity.
> **Step 4:** The player gets killed when the enemy attacks it: the game enters its game over state.

Post conditions: Enemy attempts to pursue and attack the player.
Priority: 3
ID: E03

Name: Enemy Engages the Player
Summary: The enemy engages and pursues the player.
Actors: Player, Enemy
Preconditions: Player is alive, Enemy is alive, the Enemy has been created and initialized, Enemy has noticed the player and is in an attack state.
Basic sequence:

> **Step 1:** The enemy pursues the player to get in a suitable position to attack (gets closer so it is in range or to attack with a melee weapon).
> **Step 2:** The enemy will continue to attack and pursue the player until the player has been killed, the enemy has been killed, or the enemy has lost track of the player.

Exceptions:

> **Step 1, 2:** Another game entity begins attacking the enemy while the enemy is engaging the player: the enemy ignores the entity and keeps engaging the player while the player is seen.

Post conditions: Enemy attempts to kill the player.
Priority: 1
ID: E04

Name: Enemy Attacks Ranged

Summary: The enemy engages the player with a ranged attack.
Actors: Player, Enemy
Preconditions: Player is alive, Enemy is alive, the Enemy has been created and initialized, Enemy has noticed the player and is in an attack state, Enemy attacks using a ranged attack.
Basic sequence:

> **Step 1:** The enemy pursues the player to get close enough so that its ranged attack is within range.
> **Step 2:** The enemy will target the player based on its current position and calculates for movement and projectile speed.
> **Step 3:** The enemy will use its ranged attack to attack the player, recalculating after each attack.
> **Step 4:** The enemy will continue to attack and pursue the player until the player has been killed, the enemy has been killed, or the enemy has lost track of the player.

Exceptions:

> **Step 1, 2, 3, 4:** Another game entity begins attacking the enemy while the enemy is engaging the player: the enemy ignores the entity and keeps engaging the player while the player is seen.

Post conditions: Enemy attempts to kill the player with a ranged attack.
Priority: 2
ID: E05

Name: Enemy Attacks Melee
Summary: The enemy engages the player with a melee attack.
Actors: Player, Enemy
Preconditions: Player is alive, Enemy is alive, the Enemy has been created and initialized, Enemy has noticed the player and is in an attack state, Enemy attacks using a melee attack.
Basic sequence:

> **Step 1:** The enemy pursues the player to get close enough so its melee attack is within range. This is most likely right next to the player.
> **Step 2:** The enemy will ensure its attack is within range of the player.
> **Step 3:** The enemy will use its attack to strike the player, and reposition after each attack to stay in range.
> **Step 4:** The enemy will continue to attack and pursue the player until the player has been killed, the enemy has been killed, or the enemy has lost track of the player.

Exceptions:

> **Step 1, 2, 4:** The player moves to a location where the enemy cannot reach: the enemy continues to find a path to the player’s location.

Step 1, 2, 3, 4: Another entity attacks the enemy while the enemy is attacking the player: the enemy ignores this entity and continues to attack the player.

Post conditions: Enemy attempts to kill the player with a melee attack.
Priority: 2
ID: E06

Name: Enemy is Killed
Summary: The enemy is killed.
Actors: Enemy
Preconditions: Enemy is alive, the Enemy has been created and initialized
Basic sequence:

Step 1: The enemy suffers an attack.
Step 2: The enemy runs out of health.
Step 3: The enemy dies, playing a die animation.
Step 4: The enemy drops a reward for the player.
Step 5: The enemy's body disappears after a few seconds have passed.
Step 6: The enemy's data is removed from the game and deleted.

Exceptions:

Step 1: The enemy is already dead and suffers another attack while its corpse is present: the enemy does nothing to respond.
Step 3: The enemy is in the death animation and suffers an attack: the enemy does not respond.

Post conditions: Enemy dies and is removed from the game.
Priority: 1
ID: E07

Name: Player is Killed
Summary: The player is killed.
Actors: Player
Preconditions: Player is alive
Basic sequence:

Step 1: The player suffers an attack.
Step 2: The player runs out of health.
Step 3: The player dies and plays a death animation.
Step 4: The game sets its state to a game over state.
Step 5: A game over screen opens with options to restart or go to the start menu.

Exceptions:

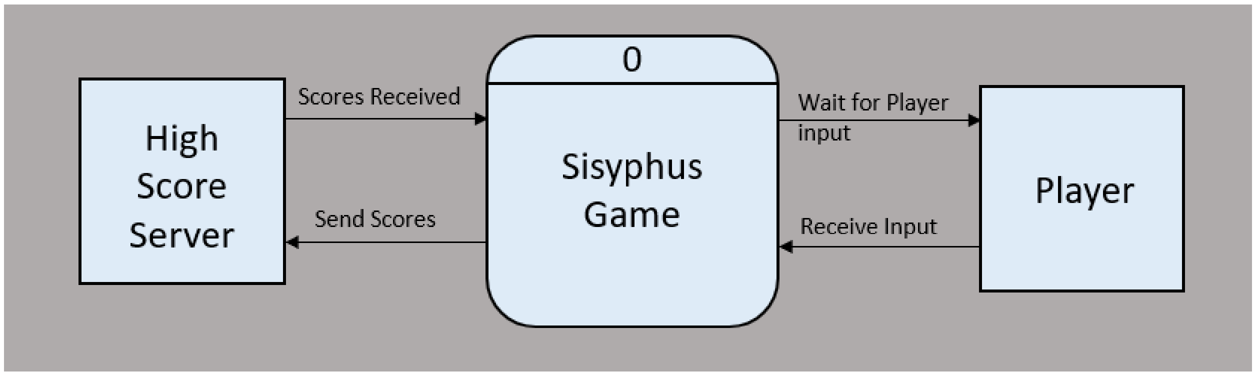
Step 3: The player attempts to heal during the death animation: the player has already died and does not respond.

Step 3: The player is in the death animation and suffers an attack: the player does not respond.

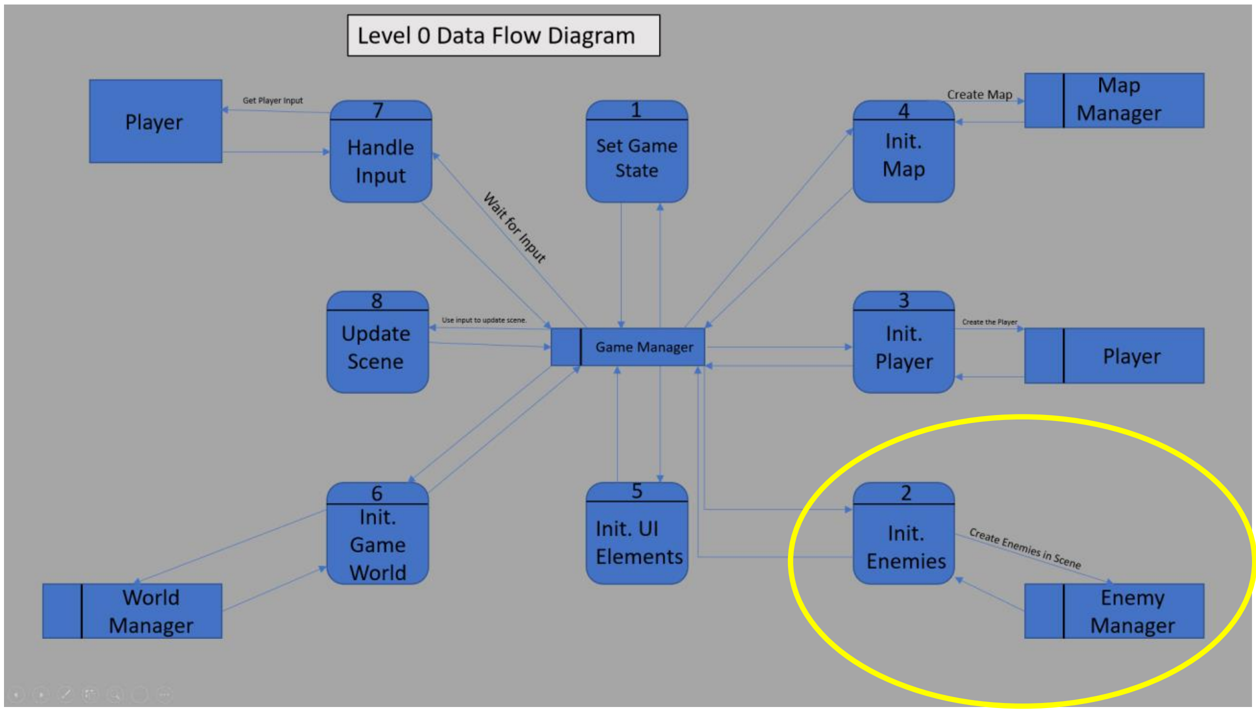
Post conditions: The player is killed; the game is set to a game over state.
Priority: 1
ID: E07

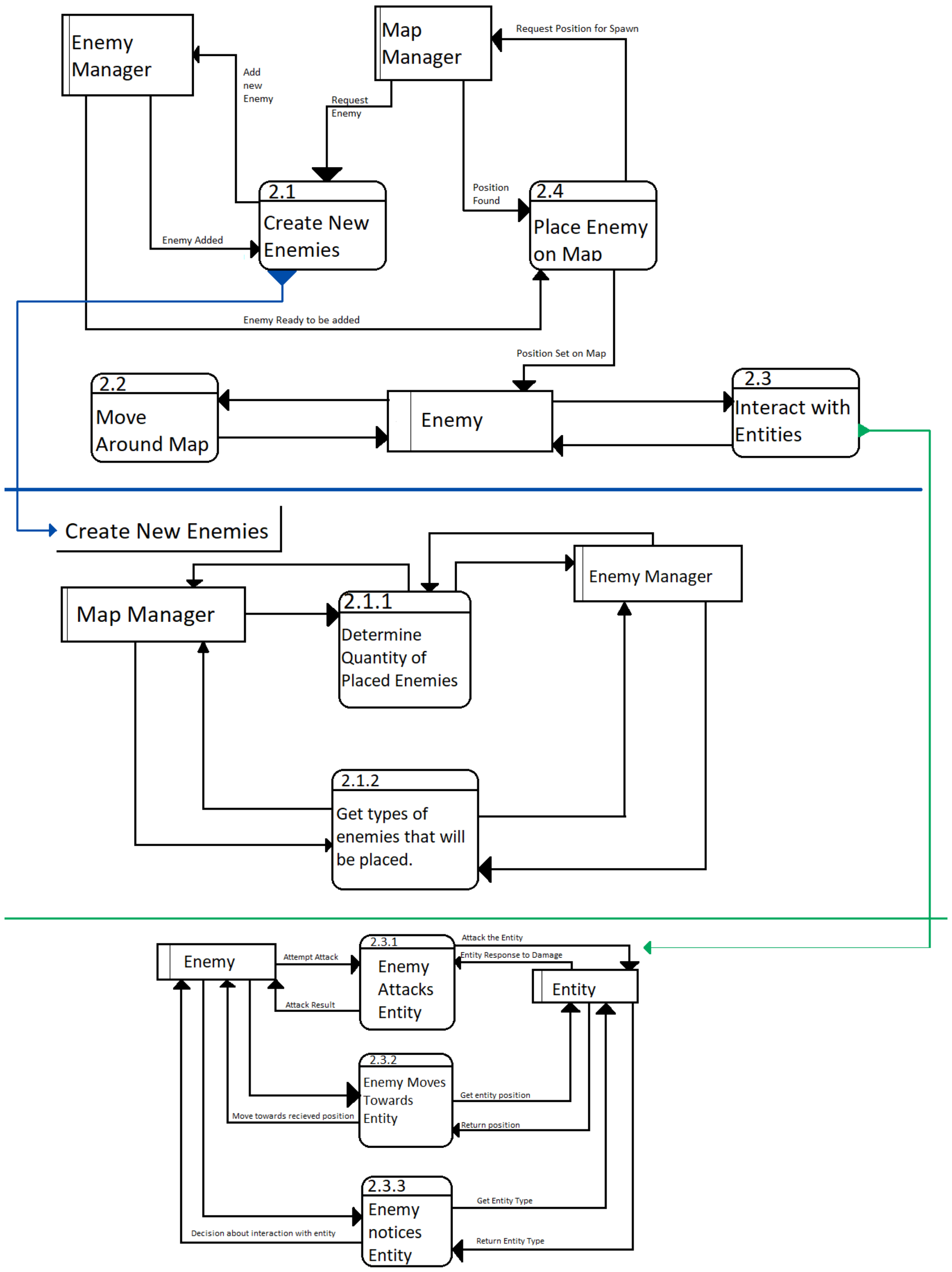
3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Context Diagram



Data Flow Diagrams




Enemy Manager
Map Manager
Request Position for Spawn
Add new Enemy
Request Enemy
2.1 Create New Enemies
Enemy Added
Position Found
2.4 Place Enemy on Map
Enemy Ready to be added
Position Set on Map
2.2 Move Around Map
Enemy
2.3 Interact with Entities
Create New Enemies
Enemy Manager
Map Manager
2.1.1 Determine Quantity of Placed Enemies
2.1.2 Get types of enemies that will be placed.
Enemy
Attempt Attack
Attack Result
2.3.1 Enemy Attacks Entity
Attack the Entity
Entity Response to Damage
Entity
2.3.2 Enemy Moves Towards Entity
Move towards recieved position
Get entity position
Return position
2.3.3 Enemy notices Entity
Get Entity Type
Return Entity Type
Decision about interaction with entity

Process Descriptions

```
2.1.2 Get Types of enemies that will be placed
    GET the map type from the Map Manager
    GET a list of possible enemies from this map type
    PASS this list to the Enemy Manager to generate enemies
    RETURN the quantity of enemies made by the Enemy manager
2.2 Move Around Map
    IF entity has not moved yet, then
        IF a movement waypoint has not been assigned, then
            GET a description of the local area from Map Manager
            Create a waypoint within this region
        ENDIF
        Move towards the movement waypoint
            Calculate the horizontal and vertical distance
            Determine whether the entity needs to jump to move
    ENDIF
2.3.1 Enemy Attacks Entity:
    IF enemy attack type is ranged then
        IF entity is within range of ranged attack, then
            Fire ranged attack at enemy position
            Create new ranged projectile
        ENDIF
        ELSE
            Move towards entity on next move, cannot attack
        ENDELSE
    ENDIF
    IF enemy attack type is melee
        IF within range of entity for melee attack, then
            Attack the entity
        ENDIF
        ELSEIF not within range of entity for melee attack, then
            Move towards entity on next move, cannot attack
        ENDELSEIF
    ENDIF

2.3.2 Enemy Moves Towards Entity
    GET the entity's position and save it
    Calculate the distance between the enemy and the entity.
    Create a 2D vector that will be added to the enemy's current velocity
    based on the current acceleration.
    IF there is a vertical difference in positions, then
        GET the local map information from the Map Manager
        Find the first vertical cell difference.
        IF we need to jump to move forward, then
```

```
                Add a jump vector to the enemy
            ENDIF
        ENDIF
        Apply the movement vector to this enemy.
```

2.3.3 Enemy Notices Entity

```
        IF entity is within a minimal notice distance from the enemy, then
            Enemy notices the entity
            Save this noticed entity as the most recently noticed entity.
        ENDIF
        IF entity is in line of sight of the enemy and the entity is within line of
        sight range, then
            Enemy notices the entity
            Save this noticed entity as the most recently noticed entity.
        ENDIF
        IF this enemy was attacked by the entity, then
            Enter an attack state immediately.
            Save the entity as the targeted entity for attack
            Remove the entity as the most recently noticed
        ENDIF
        IF a noticed entity was found and the noticed entity is attackable, and
        hostile to this enemy, then
            Enter an attack state
            Save the entity as the targeted entity for attack
            Remove the entity as the most recently noticed
        ENDIF
        ELSEIF a noticed entity was found
            Ignore the noticed entity
            Remove the entity as the noticed entity
        ENDELSEIF
```

2.4 Place Enemy on Map

```
        GET the list of generated enemies from the Enemy Manager
        WHILE there are still enemies in this list
            Grab the next enemy in the list
            Decide on a random x coordinate
            GET the y position of this selected x coordinate
            WHILE selected position is occupied
                Search both to the left and right cells for an opening
            ENDWHILE
            SET the enemies position at this location in the map
        ENDWHILE
```

4. Acceptance Tests _______9

1. Testing Enemy Pathfinding and Movement Across the Map.
    a. Create a random map piece from the map management system and place this piece in the game environment. This is the only map piece present.
    b. Create a single enemy and place it at the furthest left in this map available.
    c. Then set its movement waypoint to the furthest right point on the screen.
    d. Give the enemy one minute to get to its waypoint. If it fails to reach it, then we will record some data to help understand what went wrong. First, a screenshot will be taken and saved of the map piece. Then the following will be saved in a text file named waypointErrorX (X is the number starting at 0 of this specific error)
        i. Data about the map piece used such as seed, and other data needed to recreate this exact piece.
        ii. Data about the enemy, its type, and its position when it ran out of time.
        iii. Is the enemy still on the map, or did it fall out somehow?
    e. This process should be repeated autonomously at least 50 times.
    f. In order to pass, there must only be up to 5 instances where the enemy could not reach the waypoint. There must be no instances where the enemy fell out of the map.
2. Testing Performance of Enemy Ranged Attack Projectiles.
    a. Created an isolated map section, most likely just a flat ground layer for the enemies to stand on. Put boundaries of some kind around this map section to keep enemies within it.
    b. Create an entity that the enemies will attempt to attack above this map section. This entity will handle collisions for the projectiles, but it will be invulnerable and take not damage.
    c. Then, generate a new ranged attack enemy (all the same kind) every second until the game performance averages below 30 frames. Measure the number of this specific kind of enemies that were generated.
    d. Repeat this process at least 3 times for each of the different enemy kinds present in the game. Record this data.
3. Testing Performance of pathfinding.
    a. This scenario starts similarly to the first test.
    b. Create a random map piece from the map management system and place this piece in the game environment. This is the only map piece present.
    c. Create a single enemy and place it at the furthest left in this map available.
    d. Then set its movement waypoint to the furthest right point on the screen.
    e. Wait 30 seconds.
    f. Then repeat this process, this time adding an additional enemy. So, for the second iteration, we have two enemies trying to find the path to the same waypoint.

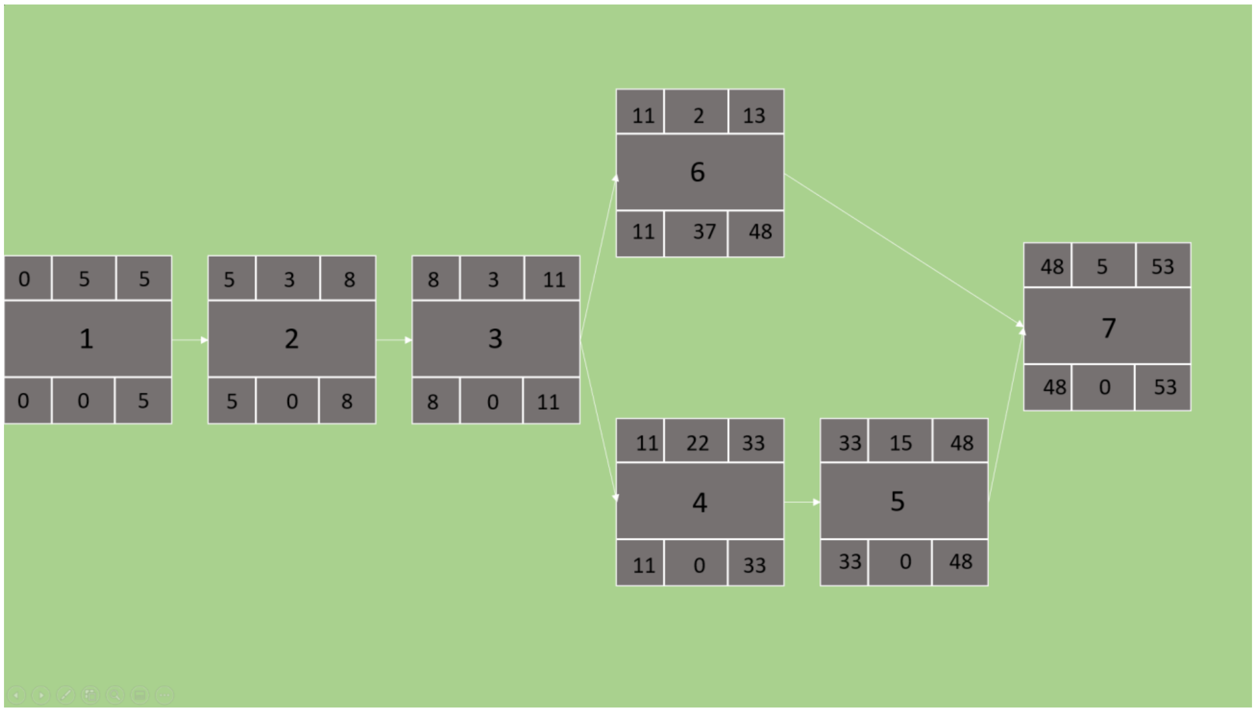
g. Repeat this for the same map piece for as long as possible until the game performance drops below 30 frames.
h. Then record how many enemies were present.
i. This process should be repeated for 50 different map pieces to get an idea of the pathfinding performance of the enemy AI.

5. Timeline ________/10

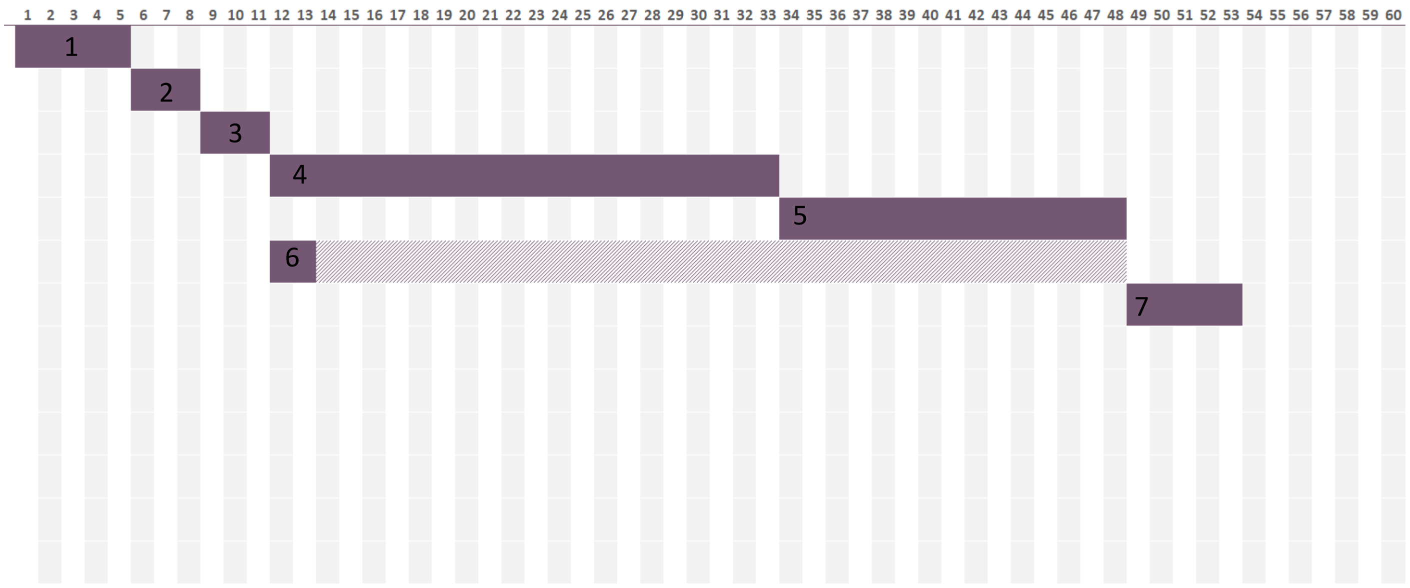
Work items

Task	Duration (hours)	Predecessor Task(s)
1. Gather all Specifics and Requirements: Initial Design	5	-
2. Finalize design with system other components	3	1
3. Report Design to Group and revise design.	3	2
4. Implementation and Programming	22	3
5. Testing and Verification	15	4
6. Documentation	2	3
7. Integration into main branch	5	4, 5, 6

Pert diagram



Gantt timeline



The Gantt chart above is indexed at one instead of 0, but it conveys the same information as the PERT above.