Name___Lucien Lee_____________ Mark ______________________/50

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

1. Brief introduction __/3

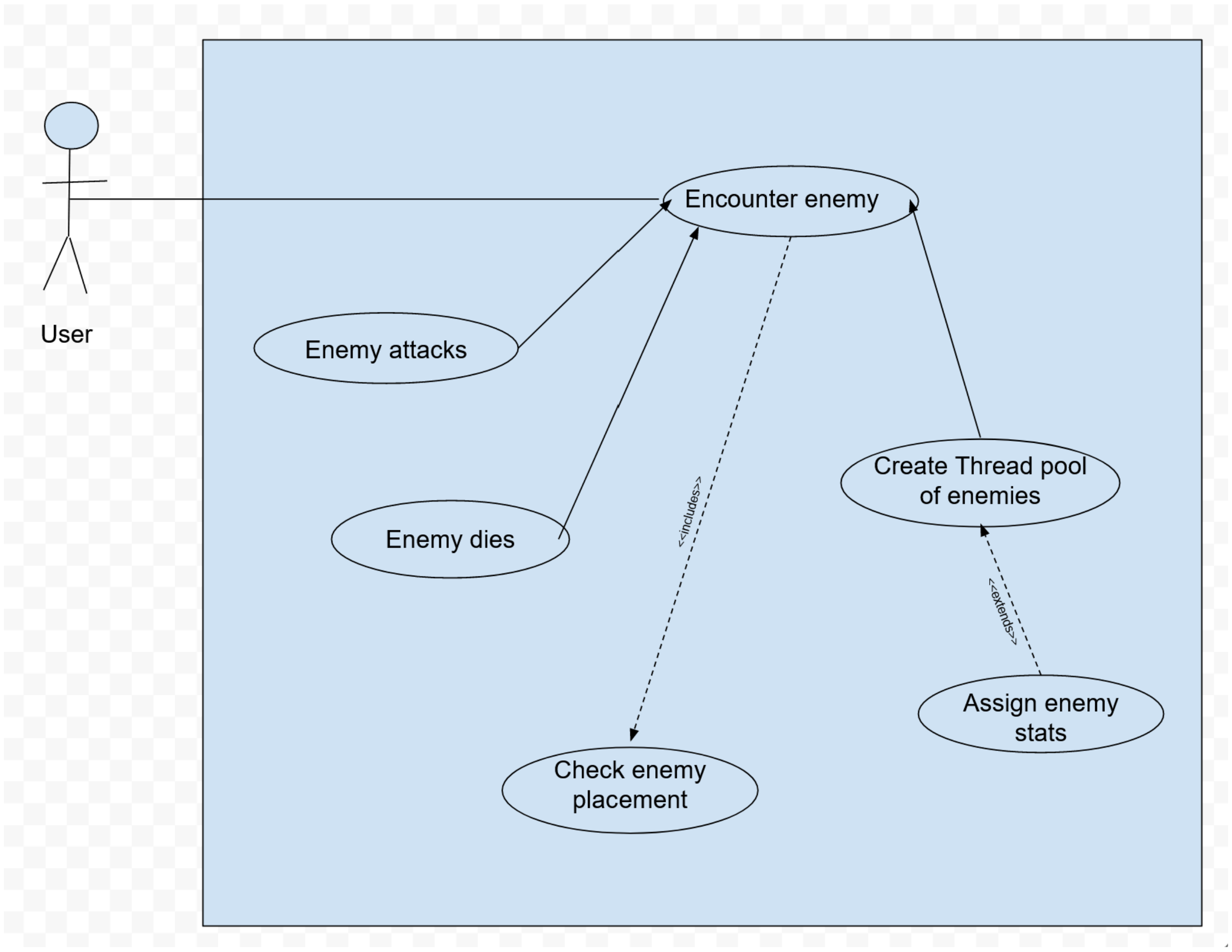
The feature that I will be creating for our video game, Exodus by Exodus, is the various enemies that will be included in the video game and how they function in the game.

When the user encounters an enemy, it is my responsibility to pull the proper enemy out of the pool of potential enemies and allow the player to attempt to defeat the enemy that is pulled. I also need to properly get rid of the enemy that is generated once they player defeats it.

This will be done with a thread pool where the enemies are generated during game downtime so they can be easily introduced to the game at the proper times. This will allow the game to not have loading screens as the game will not have to stop and reload new enemies, but rather it can constantly generate new enemies.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

Scenario 1:

Name: Encounter enemy

Summary: The user encounters an enemy in the game while controlling the player.

Actors: User

Preconditions: The player and the level have both been initialized already.

Basic sequence:

- **Step 1:** The user will encounter an enemy at a time specified by the level generator.
- **Step 2:** The stats of the enemy will be shared with the thread pool.
- **Step 3:** The thread pool will create a pool of enemies during time in the game that the system isn't too busy.
- **Step 4:** The placement of the enemy on the main level is checked to ensure proper placement.
- **Step 5:** The enemy attacks the user.
- **Step 6:** The enemy is defeated (or dies).

Exceptions:

- **Step 1:** The system never has time to make enemies for the user to encounter.
- **Step 2:** The placement of the enemy is incorrect due to the terrain that the user is currently on.
- **Step 3:** The enemy is unable to attack.
- **Step 4:** The enemy kills the user and is unable to die.

Post conditions: The enemy is successfully generated and is properly placed on the screen ready for the user to encounter.

Priority: 2*

ID: Ee1

*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

In the dataflow diagrams below, I will be covering the Create enemies feature in entirety. I will describe the "Generate enemy stats" sub-process with a decision tree.

Data Flow Diagrams

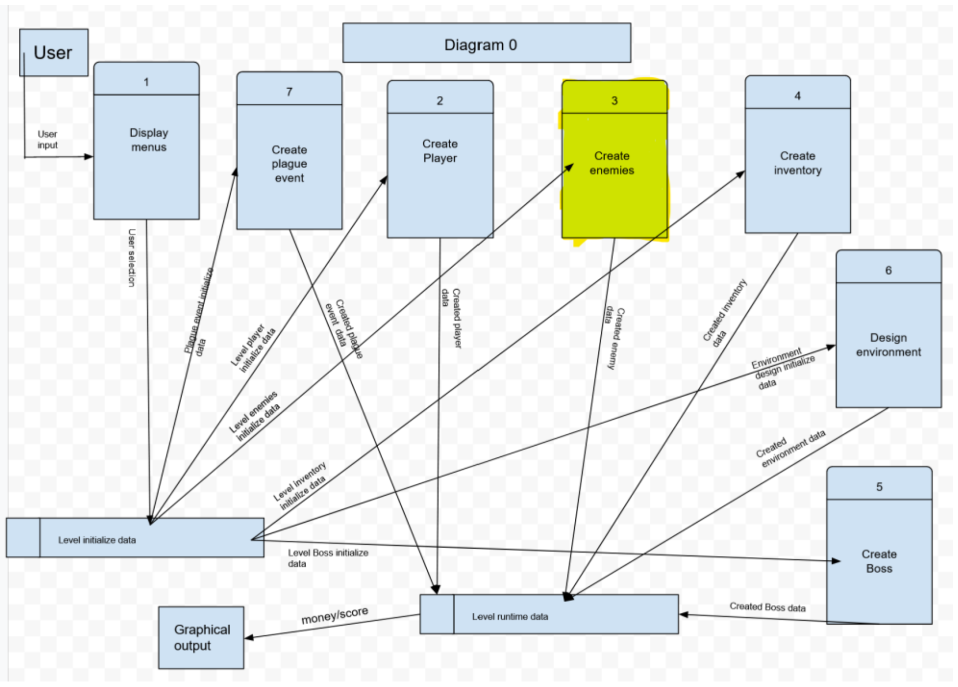


Diagram 0
User
User input
1
Display menus
7
Create plague event
2
Create Player
3
Create enemies
4
Create inventory
6
Design environment
5
Create Boss
User selection
Plague event initialize data
Level player initialize data
Level enemies initialize data
Level inventory initialize data
Environment design initialize data
Level Boss initialize data
Created plague event data
Created player data
Created enemy data
Created inventory data
Created environment data
Created Boss data
Level initialize data
Level runtime data
money/score
Graphical output

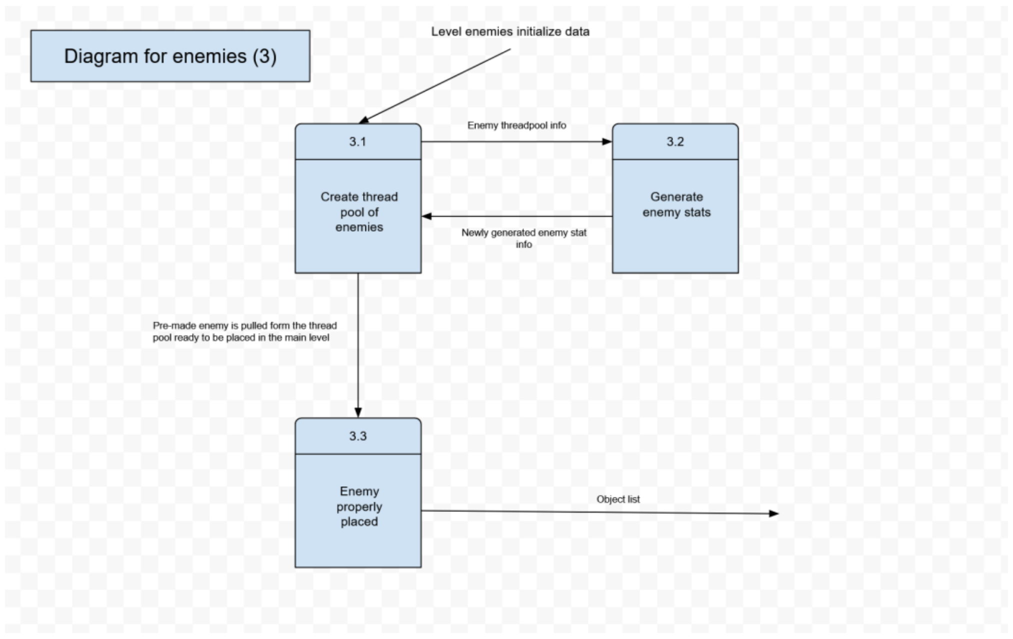
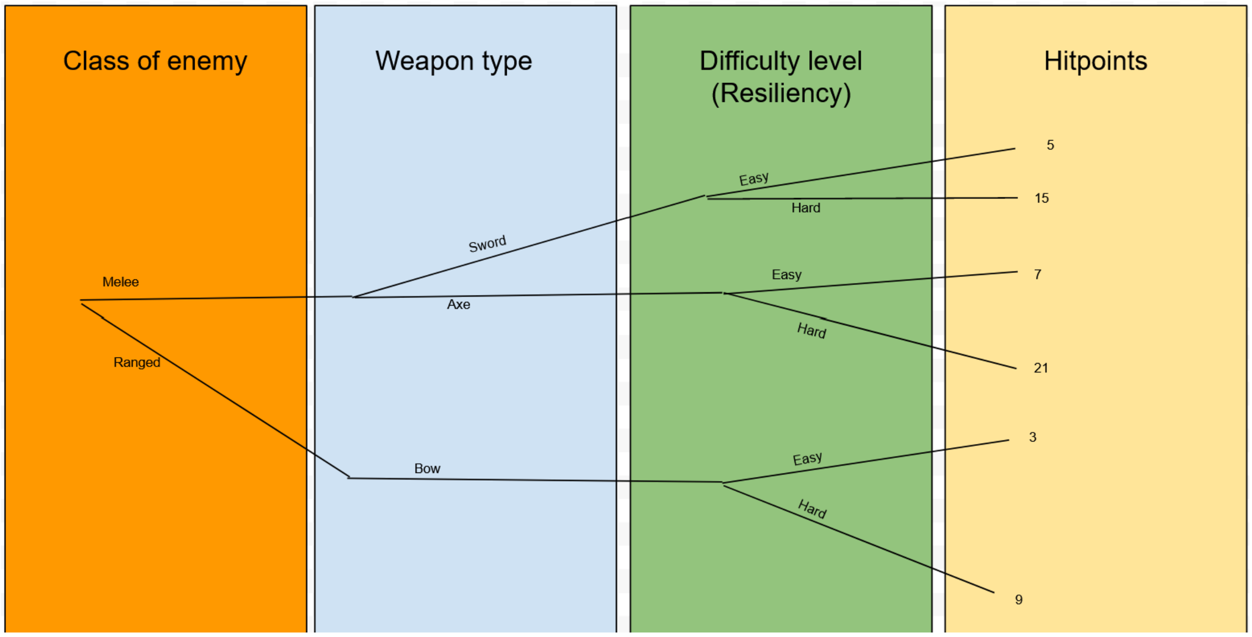


Diagram for enemies (3)
Level enemies initialize data
3.1
Create thread pool of enemies
Enemy threadpool info
3.2
Generate enemy stats
Newly generated enemy stat info
Pre-made enemy is pulled form the thread pool ready to be placed in the main level
3.3
Enemy properly placed
Object list

Process Descriptions

The process description for Process 3.2 is displayed below in a decision tree.



4. Acceptance Tests ______9

This feature has multiple predeveloped elements such as the enemy type and the corresponding stats for that enemy. This feature also has some random elements applied to it such as where the enemies will spawn and what type of enemy will be spawned at a certain location.

This feature needs to be tested as to whether the enemies that are generated are a mixture of both hard enemies and easy enemies and if the class of enemy that is chosen is truly random or not.

The acceptance test for this feature is shown below.

Enemy stat assigner:

This feature will be run 1000 times checking each type of enemy that is created and if they are randomized between easy and difficult enemies, as well as making sure that the enemy class is diverse and there are not too many of one type of enemy.

The output file should display the type of enemy, the difficulty of the enemy, and the total number of each class of enemy and total number of easy and hard enemies once the generation program has been run.

Example output test for the generation of enemies.

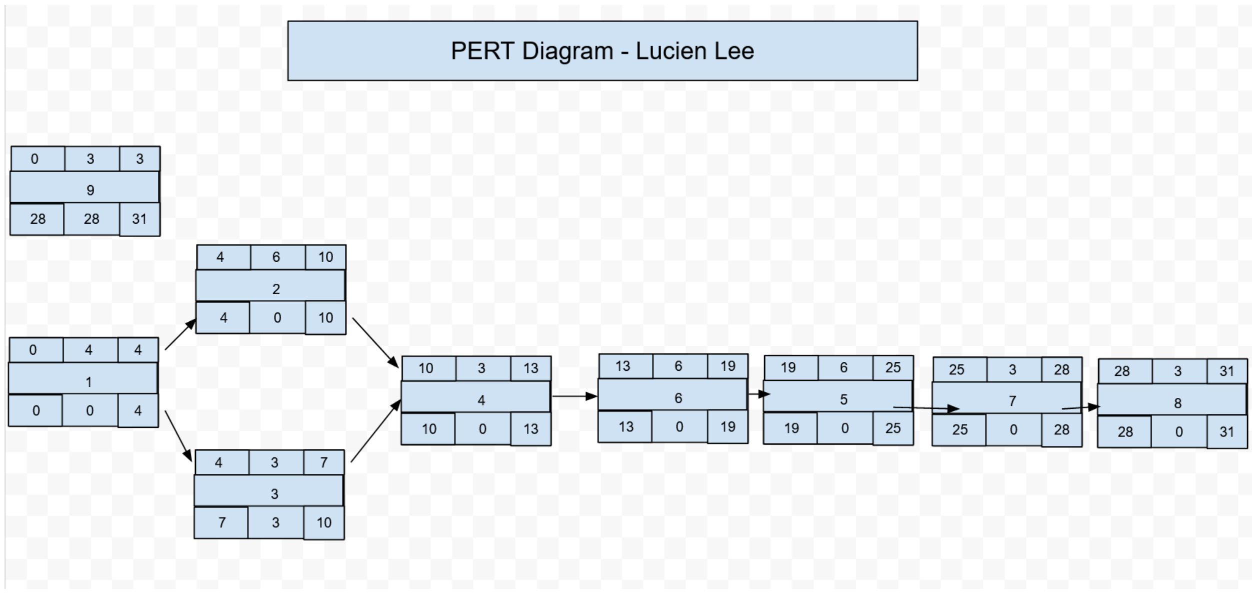
Output (number)	Class	Easy	Hard	Total melee	Total ranged	Total easy	Total hard
1	Melee	No	Yes	1	0	1	0
100	Ranged	No	Yes	32	68	48	52
500	Ranged	Yes	No	250	250	275	225
861	Melee	Yes	No	431	430	460	401
1000	Melee	No	Yes	500	500	500	500

5. Timeline _______/10

Work items

Task	Duration (Hrs)	Predecessor Task(s)
1. Enemy Design	4	-
2. Initial Enemy Function Creation/Programming	6	1
3. Thread pool Design	3	1
4. Thread pool Construction/Programming	3	2, 3
5. Enemy Placement Programming	6	4, 6
6. Programming	6	2, 4
7. Testing	3	6
8. Installation	3	7
9. Artwork	3	-

Pert diagram

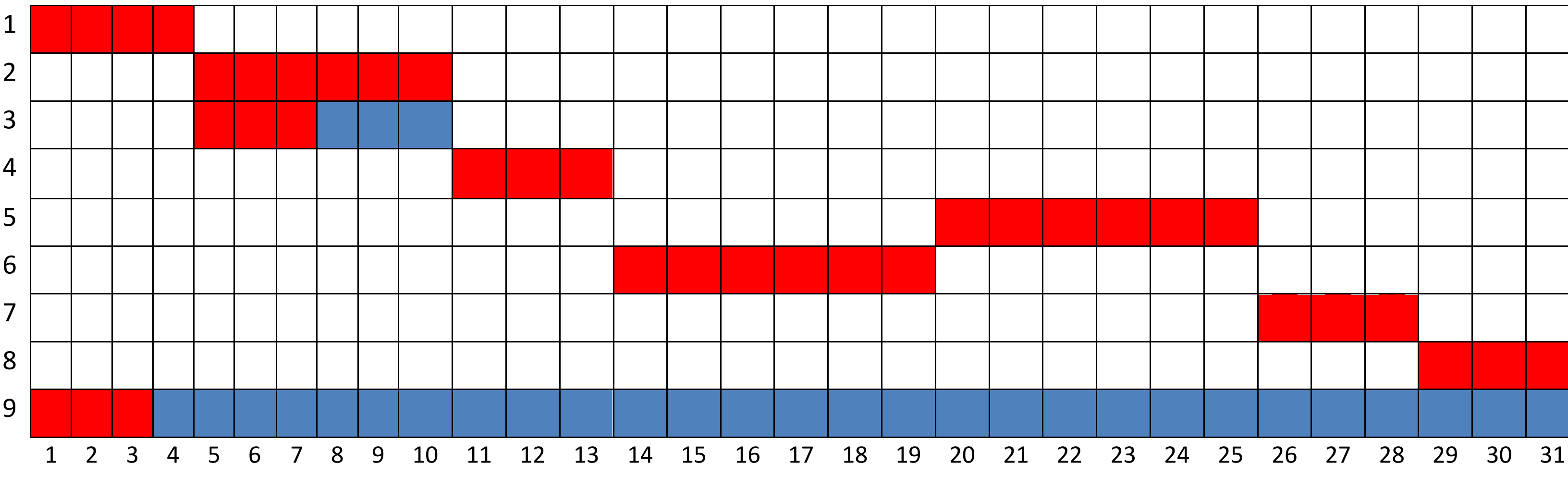


Gantt timeline

Gantt Chart – Lucien Lee

Key:

🟥	**Work Hours**
🟦	**Slack**



1
2
3
4
5
6
7
8
9
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31