Name\_Jacob Porath\_\_\_\_\_\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My feature for Coeur d’Game’s Rogue Realm will be the level generation.

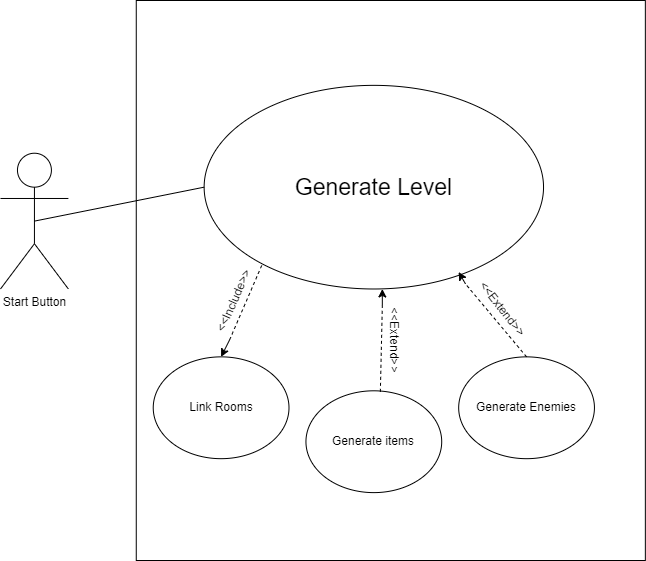
A level will be generated when the player starts the game from the main menu, and when they complete a level and exit through that level’s final door. As the player completes more levels, they will become more powerful, but the enemies will become stronger too.

The core characteristic of Rogue Realm’s level generation will be randomization. The levels will be assembled from a set of pre-made rooms. Each room will have doors in and out of them. When a level is generated, the doors will be linked together in random but controlled patterns to create unique environments for the player to progress through.

Once the level’s rooms are linked, they will be populated with enemies to challenge the player and items to make the player more powerful. These too will be random. Enemies and items will be randomly selected from a pool of possible options. Sometimes the player may enter a room and find nothing at all. This should be rare, but randomization does allow the possibility. This random generation will serve to make each play through of the game unique.

## Use case diagram with scenario \_\_14

### Use Case Diagrams



### Scenarios

**Name:** Start the game

**Summary:** The player clicks the “Start” button on the main menu and a new level is generated to begin the game.

**Actors:** Start button

**Preconditions:** None

**Basic sequence:**

**Step 1:** Link starting and ending rooms through a series of random rooms.

**Step 2:** complete any branching paths with additional randomly selected rooms.

**Step 3:** For each room in the level, randomly determine if enemies will be placed.

**Step 4:** For each room in the level, randomly determine if items will be placed.

**Exceptions:**

**Step 1:** If an enemy is placed, randomly select an enemy type and place it in the level.

**Step 2:** If an item is placed, randomly select an item and place it in the level.

**Post conditions:** Starting room is displayed with the player character in it.

**Priority:** 1\*

**ID:** 2A

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

### Data Flow Diagrams

Context Diagram:

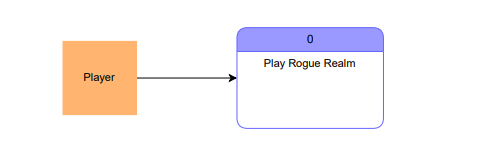


Diagram 0: (My feature is process 2)

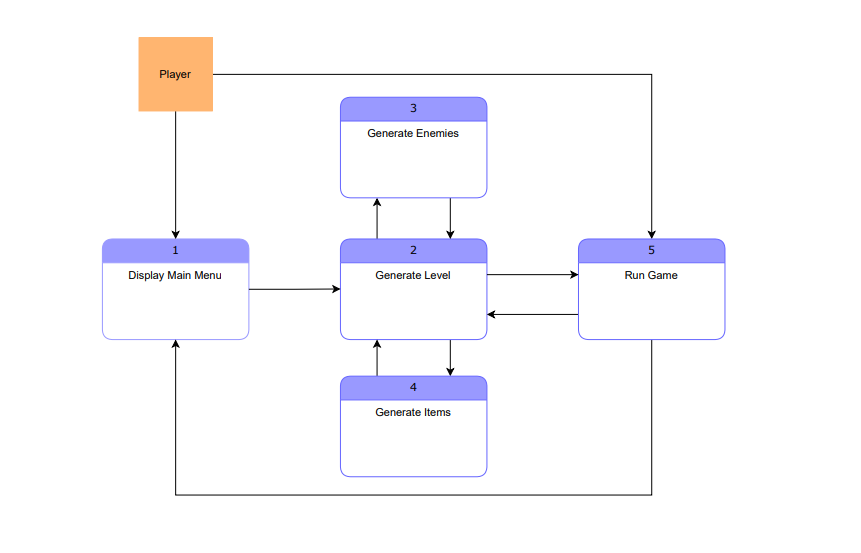
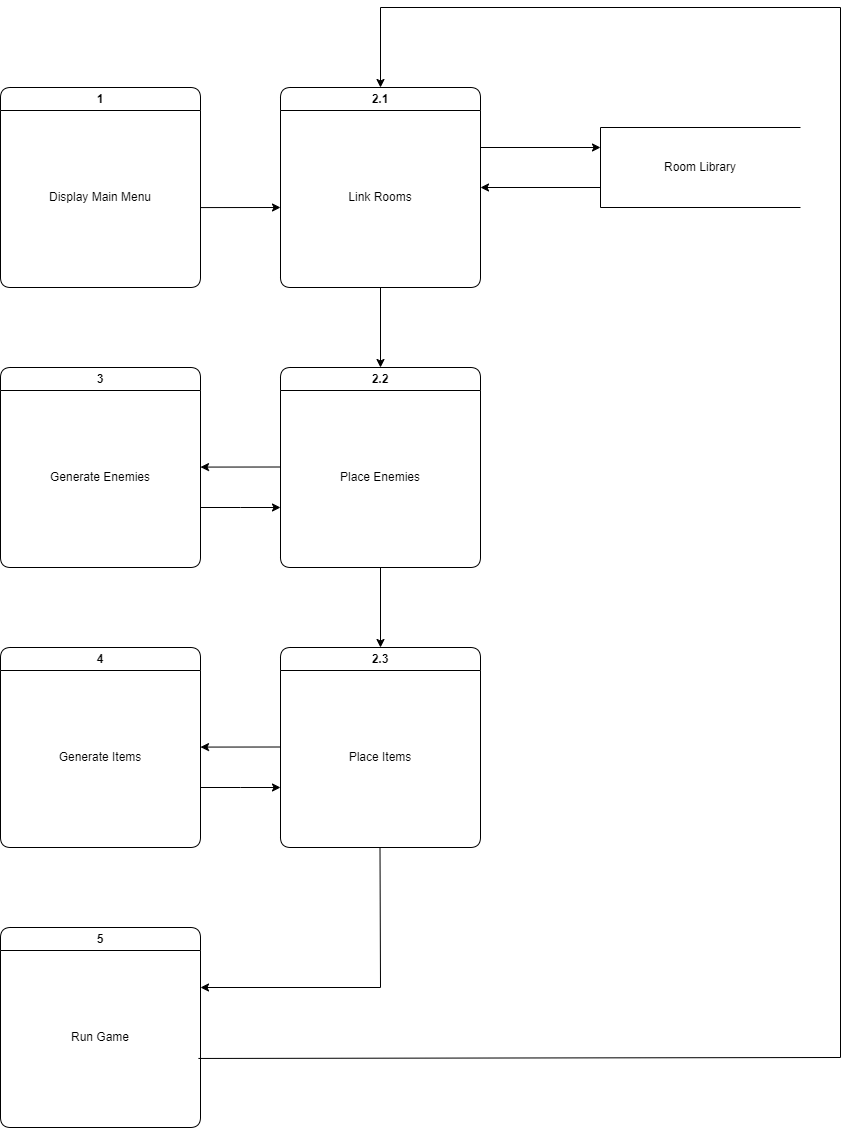


Diagram 1: (All level generation features labelled 2.x)



### Process Descriptions

Process 2.1: Link rooms

All rooms are selected randomly. A level should begin with a starting room, containing the player. Random rooms will be generated linking to an ending room. Any doors branching from this path will be filled in with branching rooms until no unlinked doors remain.

Process 2.2: Place Enemies

For each room in the level, random number generation will determine if enemies will be placed inside them. When an enemy is to be placed in a room, the type of enemy and its attributes will be given by Process 3 (Generate enemies).

Process 2.3: Place Items

For each room in the level, random number generation will determine if items will be placed inside them. When an item is to be placed in a room, the type of item and its effects will be given by Process 4 (Generate items).

## Acceptance Tests \_\_\_\_\_\_\_\_9

Use the level generation feature 10 times.

The output file will have the following characteristics:

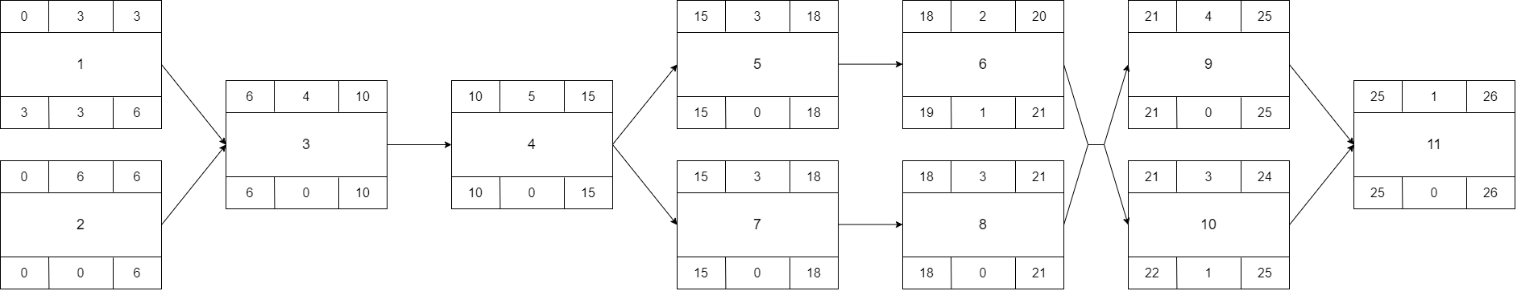
* Number of inaccessible rooms: 0 (if this is not 0, major problems exist.)
* Maximum possible number of items generated: x
* Number of items generated: y (this value should always be close to the median between 0 and x .)
* Maximum possible number of enemies generated: a
* Number of enemies generated: b (this value should always be close to the median between 0 and a .)
* Number of Boss type enemies: 10 (should always be one boss enemy per level generated.)

## Timeline \_\_\_\_\_\_\_\_\_/10

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hours) | Predecessor Task(s) |
| 1. Individual Room Design | 3 | - |
| 2. Door linking algorithm design | 6 | - |
| 3. Create Individual Rooms | 4 | 1, 2 |
| 4. Door linking algorithm programming | 5 | 3 |
| 5.Enemy placement algorithm design | 3 | 4 |
| 6.Enemy placement algorithm programming | 2 | 5 |
| 7.Item placement algorithm design | 3 | 4 |
| 8.Item placement algorithm programming | 3 | 7 |
| 9. Documentation | 4 | 6, 8 |
| 10. Testing | 3 | 6, 8 |
| 11. Installation | 1 | 10 |

### Pert diagram



### Gantt timeline

