

Champion document

Company: ScoSoft | Game: Scoto | Feature: Level generation

Name: Zachariah Preston | Date: 9/20/2021

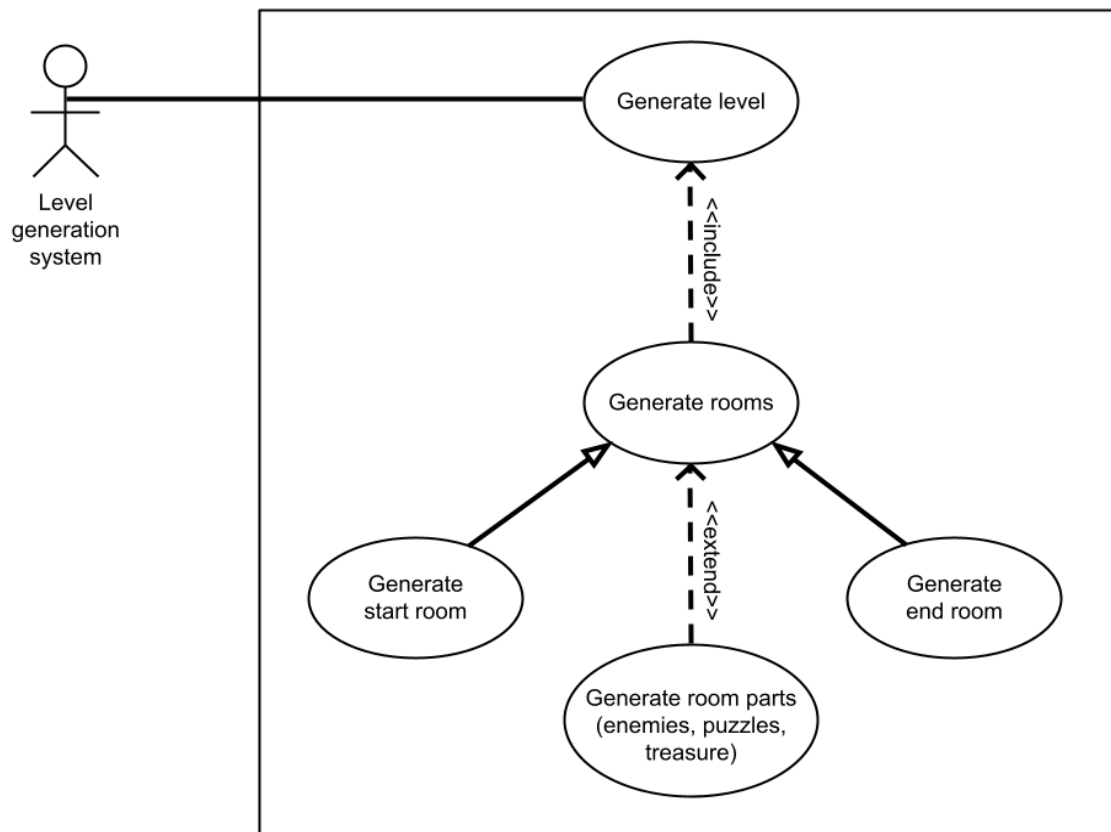
1. Brief introduction

My feature for the Scoto video game is the level generation that determines the general structure for the levels each time a new one is loaded.

The levels in Scoto will be procedurally-generated; each time a new game is started or a level is completed, a new one is created using a maze generator algorithm. The levels consist of a web of rooms. The player starts in one of the rooms, and another room (the end room) will allow the player to progress to the next level. Each room may contain enemies that the player has to avoid or fight, puzzles to solve, and treasure or items that give an advantage. (These features will be done by other team members.) The rooms will be connected to form a maze, using the maze generator algorithm as previously mentioned. The number of rooms and the parts within each room will be dependent on how many levels the player has already completed.

2. Use case diagram with scenario

Use Case Diagrams



Scenarios

Scenario 1 (first Use Case Diagram):

Name: Generate level

Summary: The level generation system creates a procedurally-generated level.

Actors: Level generation system

Preconditions: The player has just started a new game or finished a previous level.

Basic sequence:

Step 1: Create a start room.

Step 2: Branch off in a random direction and add another room. Repeat a random number of times.

Step 3: Branch off and create an end room with an exit to the next level.

Step 4: Choose a random room.

Step 5: Repeat steps 2 and 4 until the desired number of rooms is reached.

Step 6: Load the start room and put the player in it.

Exceptions:

Step 2: The new room needs features added: Load enemies, puzzles, or treasure into the room.

Post conditions: The level is created and the player can start playing it.

Priority: 1*

ID: Z01

*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

Data Flow Diagrams

Diagram 0

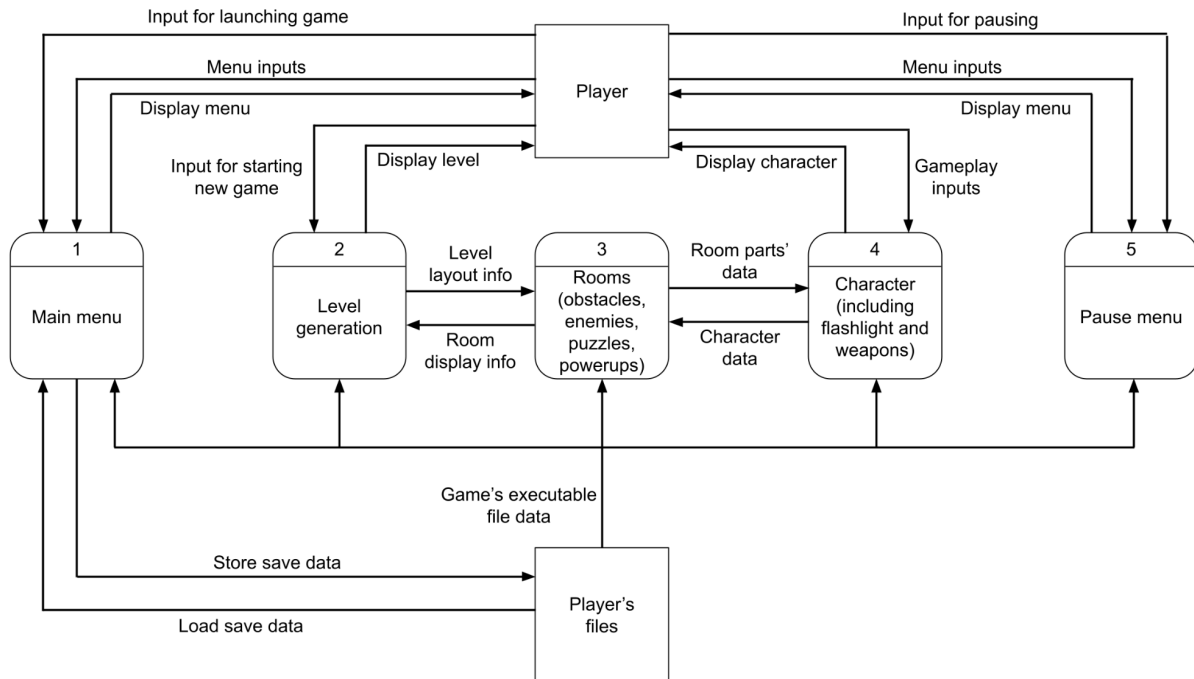
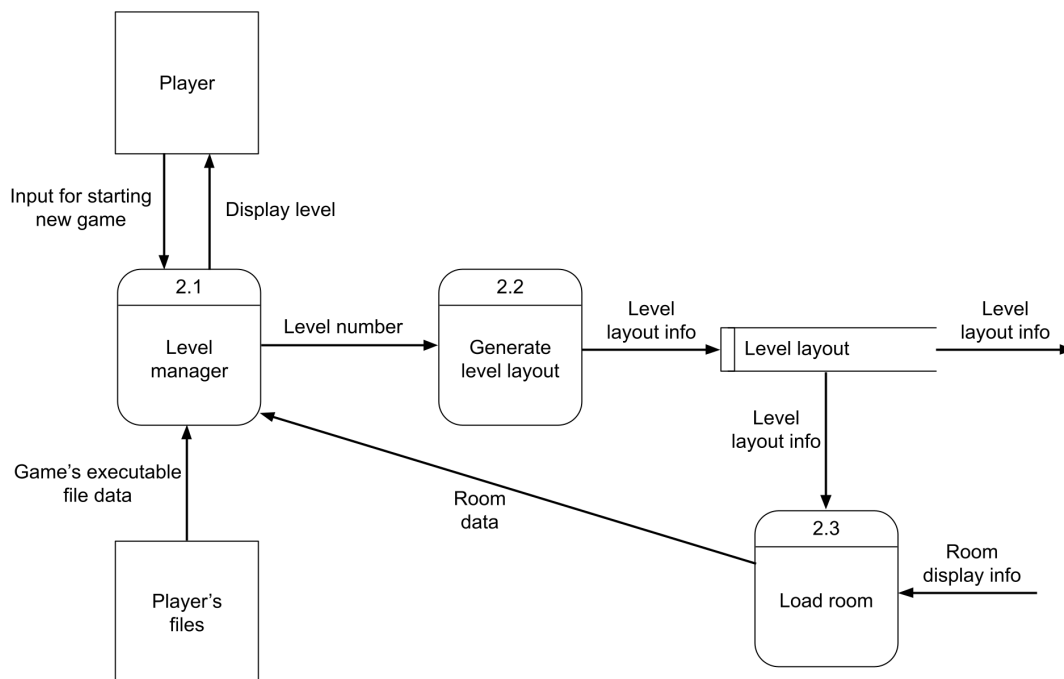


Diagram 2 – Level generation



Process Descriptions

Process 2.3

Structured English process description for “Load room”:

```
IF start room
    Load player
ELSE
    IF end room
        Load exit to next level
    END IF
    FOR each enemy in room
        Load enemy
    END FOR
    FOR each puzzle in room
        Load puzzle
    END FOR
    FOR each treasure in room
        Load treasure
    END FOR
END IF/ELSE
```

4. Acceptance Tests

Generate level layout

Test if there are too many or too few rooms, depending on the level number. For example, level 1 should have from 3 to 6 rooms.

Test if the maze is structured incorrectly. For example, make sure that there is a way to get from the start room to the end room, no rooms are impossible to reach, no two rooms are in the same spot, each room has the correct connections to adjacent ones, etc.

Load room

Test if any of the room's parts loaded incorrectly. For example, if the room type dictates that the room should have 3 enemies, check if there are more or less than 3 enemies.

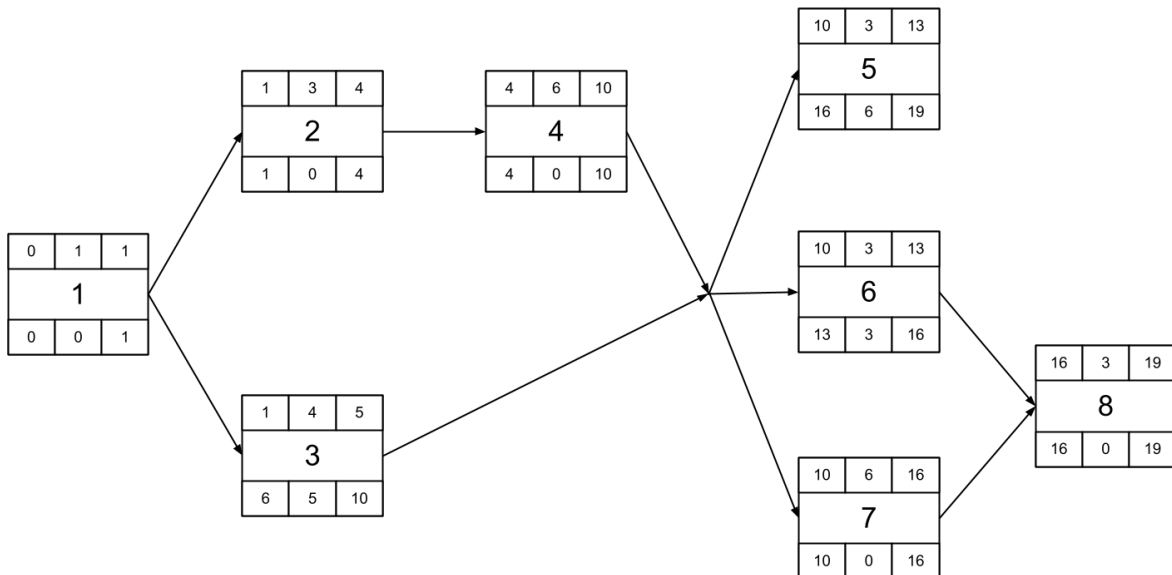
Test if the start room or end room loaded incorrectly. Make sure that the player is in the start room, the start room is empty, and the end room has an exit to the next level.

5. Timeline

Work items

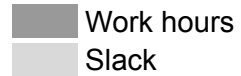
Task	Duration (pred. hours)	Predecessor task(s)
1. Layout design	1	-
2. Room designs	3	1
3. Layout programming	4	1
4. Room programming	6	2
5. Documentation	3	3, 4
6. Testing	3	3, 4
7. Artwork	6	3, 4
8. Implementation	3	6, 7









































Pert diagram



Gantt timeline

Key:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1. Layout design																			
2. Room designs																			
3. Layout prog.																			
4. Room prog.																			
5. Documentation																			
6. Testing																			
7. Artwork																			
8. Implementation																	