Name: Luke Halla Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

The feature I will be implementing is the boss fight. This will be the final battle at the end of the game that consists of unique mechanics like special abilities and new enemies. This will also include unique animations as the player weakens the boss.

## Use case diagram with scenario \_\_14

Example:

### Use Case Diagrams

A diagram of a business process

Description automatically generated

### Scenarios

**Name:** Boss Sprite

**Summary:** The new sprite created to act as the boss.This boss will be an iteration of the enemies encounter in the game. In comparison to the other enemies in the game it will have more health, deal more damage, and have unique special abilities.

**Actors:** AI

**Preconditions:** Player has made it to the last level. The base level enemies have been created.

**Basic sequence:**

**Step 1:** Check that player has made it to the final level

**Step 2:** Create scene and spawn boss

**Step 3:** Start final boss battle

**Step 4:** Player wins or loses the level

**Exceptions:**

**Step 1:** Player doesn’t make it to the boss level (dies beforehand).

**Step 2:** “BC” mode enabled for easier boss level

**Post conditions:** The winning/losing screen

**Priority:** 2\*

**ID:** C07

\*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

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

Example:

### Data Flow Diagrams



### Process Descriptions

Assign rooms\*:

WHILE teacher in two places at once OR two classes in the same room

Randomly redistribute classes

END WHILE

**\*Notes**: Yours should be much longer. You could use a decision tree or decision table instead if it is more appropriate.

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

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

**Example for random number generator feature**

Run feature 1000 times sending output to a file.

The output file will have the following characteristics:

* Max number: 9
* Min number: 0
* Each digit between 0 and 9 appears at least 50 times
* No digit between 0 and 9 appears more than 300 times
* Consider each set of 10 consecutive outputs as a substring of the entire output. No substring may appear more than 3 times.

**Example for divide feature**

|  |  |  |  |
| --- | --- | --- | --- |
| Output | Numerator  (int) | Denominator  (int) | Notes |
| 0.5 | 1 | 2 |  |
| 0.5 | 2 | 3 | We only have 1 bit precision for outputs. Round all values to the nearest .5 |
| 0.0 | 1 | 4 | At the 0.25 mark always round to the nearest whole integer |
| 1.0 | 3 | 4 | At the 0.75 mark always round to the nearest whole integer |
| 255.5 | 5 | 0 | On divide by 0, do not flag an error. Simply return our MAX\_VAL which is 255.5. |

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

[Figure out the tasks required to complete your feature]

Example:

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PWks) | Predecessor Task(s) |
| 1. Requirements Definition | 1 | - |
| 2. Screen Design | 4 | 1 |
| 3. Object Design | 4 | 1 |
| 4. User Documentation | 3 | 2,3,5,6 |
| 5. Programming | 6 | 2,3 |
| 6. Testing | 5 | 5 |

### Pert diagram

A diagram of a computer code

Description automatically generated

### Gantt timeline

