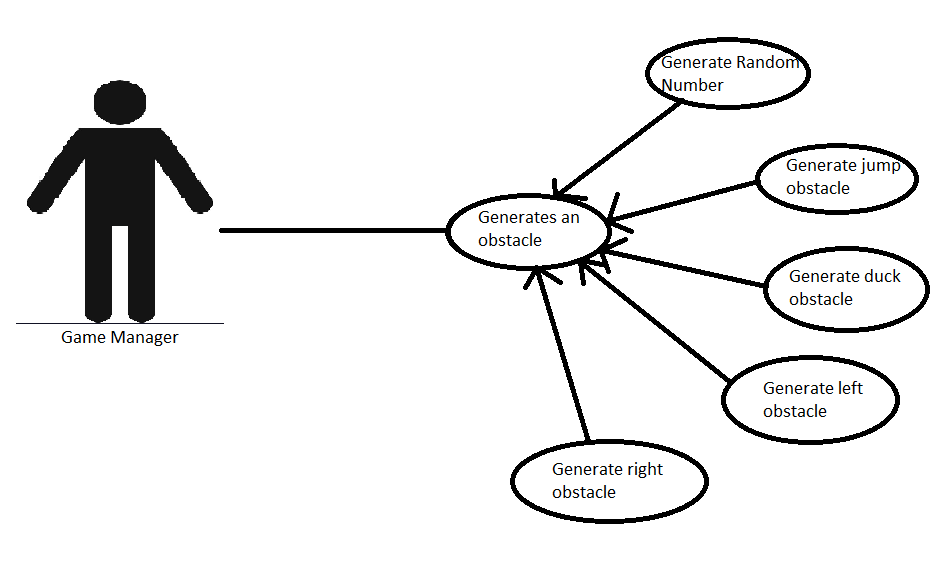
Name\_\_\_\_\_\_\_Lennin Rodriguez\_\_\_\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

I will be making the obstacles that will be going on the track of the game. We will have a jump, duck, force-you-left, and a force-you-right obstacle.

## Use case diagram with scenario \_\_14

### Use Case Diagrams



Music Manager

<<Extends>>

Coins

### Scenarios

**Name:** Generate Random Number

**Summary:** Randomly generates a number between 1 and n. n being the number of obstacles we have.

**Actors:** Music Manager

**Preconditions:** Game Manager sends the signal to generate an obstacle

**Basic sequence:**

**Step 1:** Receive signal from Game Manager

**Step 2:** Generate number between 1 and n

**Step 3:** Send number to Obstacle Generator

**Post conditions:** Calculated value is displayed.

**Priority:** 1

**Name:** Generate Jump Obstacle

**Summary:** Generates the jump obstacle

**Actors:** Music Manager

**Preconditions:** The number that Generate Random Number outputs corresponds to the id of the jump obstacle

**Basic sequence:**

**Step 1:** Receive number from Random Number Generator

**Step 2:** Match number to id of obstacle

**Step 3:** Initialize the obstacle

**Step 4:** Obstacle Generator then sends the obstacle to Game Manager

**Post conditions:** Generated obstacle appears on track

**Priority:** 1

**Name:** Generate Duck Obstacle

**Summary:** Generates the duck obstacle

**Actors:** Music Manager

**Preconditions:** The number that Generate Random Number outputs corresponds to the id of the duck obstacle

**Basic sequence:**

**Step 1:** Receive number from Random Number Generator

**Step 2:** Match number to id of obstacle

**Step 3:** Initialize the obstacle

**Step 4:** Obstacle Generator then sends the obstacle to Game Manager

**Post conditions:** Generated obstacle appears on track

**Priority:** 1

**Name:** Generate Left Obstacle

**Summary:** Generates the left obstacle

**Actors:** Music Manager

**Preconditions:** The number that Generate Random Number outputs corresponds to the id of the left obstacle

**Basic sequence:**

**Step 1:** Receive number from Random Number Generator

**Step 2:** Match number to id of obstacle

**Step 3:** Initialize the obstacle

**Step 4:** Obstacle Generator then sends the obstacle to Game Manager

**Post conditions:** Generated obstacle appears on track

**Priority:** 1

**Name:** Generate Right Obstacle

**Summary:** Generates the right obstacle

**Actors:** Music Manager

**Preconditions:** The number that Generate Random Number outputs corresponds to the id of the right obstacle

**Basic sequence:**

**Step 1:** Receive number from Random Number Generator

**Step 2:** Match number to id of obstacle

**Step 3:** Initialize the obstacle

**Step 4:** Obstacle Generator then sends the obstacle to Game Manager

**Post conditions:** Generated obstacle appears on track

**Priority:** 1

**Name:** Generate Coins

**Summary:** Generates coins for the game

**Actors:** Music Manager

**Preconditions:** Is given the signal by the music manager to produce coins for the track.

**Basic sequence:**

**Step 1:** Receive signal from Random Number Generator

**Step 2:** generate coins

**Step 3:** Initialize the coins

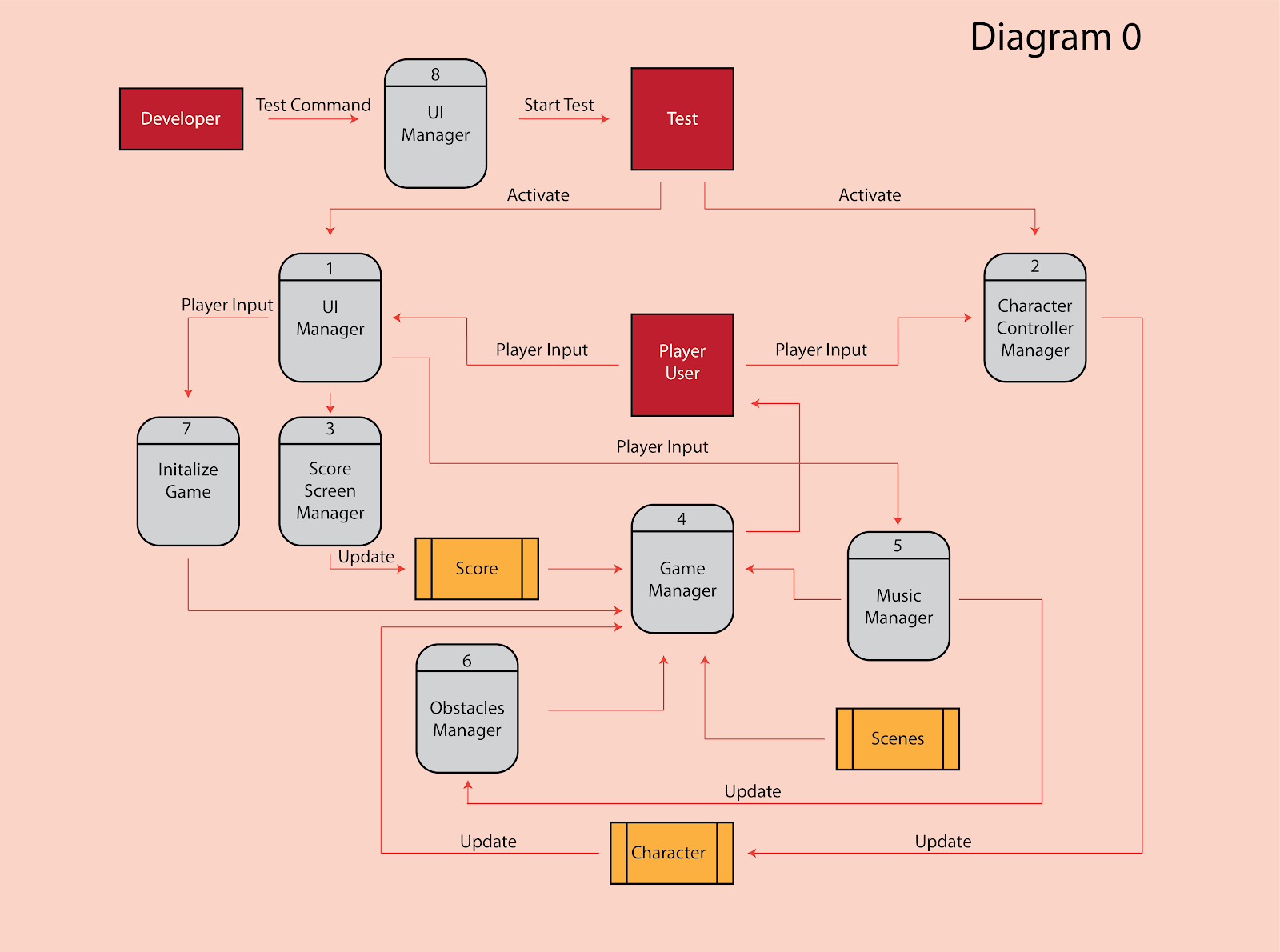
**Step 4:** Obstacle Generator then sends the coins to Game Manager

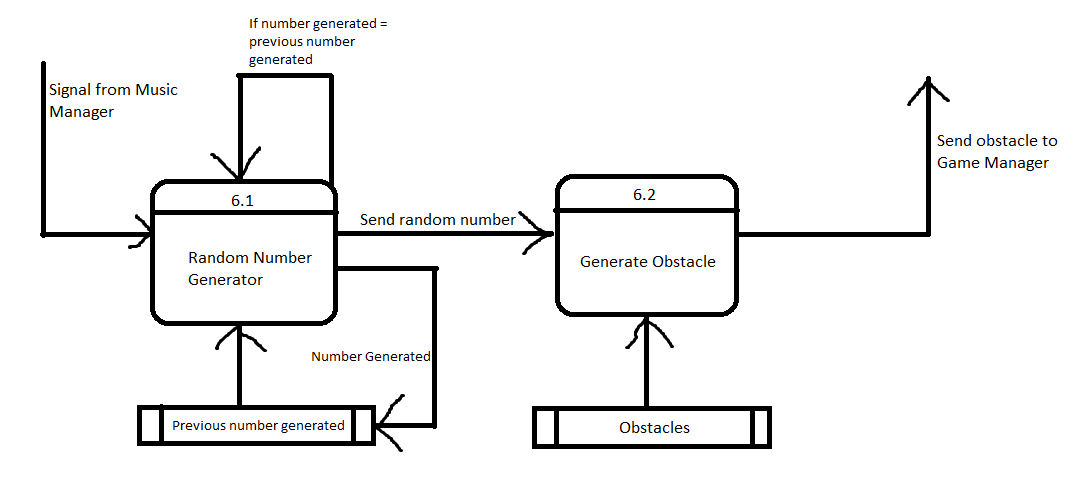
**Post conditions:** Generated coins appear on track

**Priority:** 1

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

### Data Flow Diagrams





### Process Descriptions

Random Number Generator(RNG)\*:

This process will randomly generate a number between 1 and 4 and it will send that number to the “Generate Obstacle” process.

Sample code looks like

Random r = new Random();

int rInt = r.Next(0,4);

Generate Obstacle\*:

This process will match the random number received from the RNG to the id of the obstacles in storage. After it finds the match, it will send that obstacle info to be initialized by the game manager.

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

Run obstacle generation 1000 times sending output to a file.

The output file will have the following characteristics:

* Max number: 4
* Min number: 0
* Each number that it generates, make sure that it can identify the obstacle that corresponds to it
* Make sure we can get it to match the number with the obstacle 1000 times without fail

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

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PWks) | Predecessor Task(s) |
| 1. Obstacle Design | 5 | - |
| 2. Obstacle Implementation | 10 | 1 |
| 3. Programming | 20 | 2 |
| 4. Testing | 10 | 3 |
| 5. Installation | 5 | 4 |

### Pert diagram



### Gantt timeline

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |  |
|  | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 | 33 | 36 | 39 | 42 | 45 | 48 | 51 |