Name: Benjamin Hallman Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

The feature that I will be handling is player management. I receive direct input from the user to streamline the process. I am responsible for moving the player and checking its possible updated position using collision checks. Ensuring the player’s movements are smooth, they cannot travel out of bounds, and that collision detection works are some of the outlier functions that I will be implementing.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

Move Forward

Attack

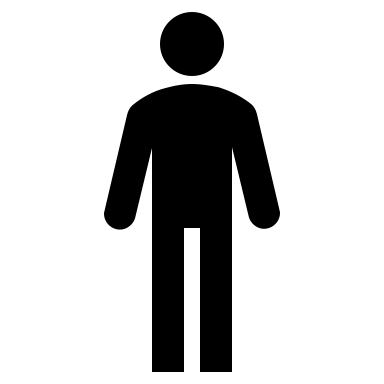
<<extend>>

<<extend>>

<<extend>>

Move Backward

Move Player



<<extend>>

Move Right

<<extend>>

Move Left

Resume

Player

<<extend>>

Pause Game

View Controls

<<extend>>

### Scenarios

**Name:** Move Player

**Summary:** Moves the player around the world.

**Actors:** Player

**Preconditions:** The game has started and the player is in a legal area.

**Basic sequence:**

**Step 1:** Recieve player control input.

**Step 2:** Based on the input from the player, produce visual and audio feedback.

**Step 3:** At anytime the player may pause the game.

**Exceptions:**

**Step 1:** Wall collision detected: Set Position to Previous Location  
 **Step 2:** Player attempts to attack without a weapon: Do Nothing

**Step 3:** [Pause Game] is active and another action is attempted other than one present on the pause menu: Do Nothing

**Post conditions:** Player receives visual update based on input.

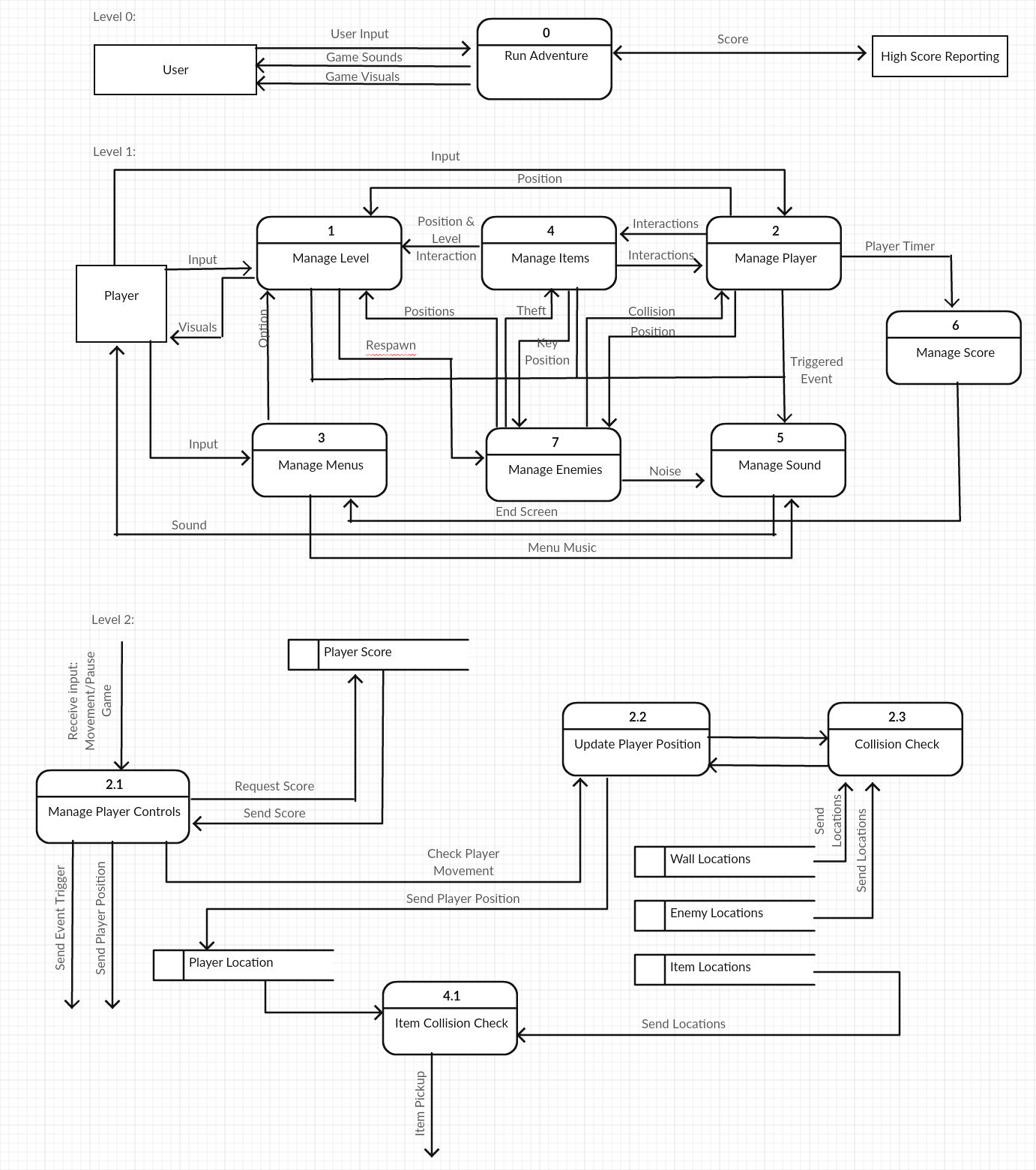
**Priority:** 1

**ID:** BH2

\*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



### Process Descriptions

Manage Player Controls:

IF player controls receives input THEN:

IF input is movement THEN:

Check player movement with update player position

ELSE:

Send event trigger

ENDIF

ENDIF

Update Player Position:

IF update player position is called to check player movement THEN:

Send the updated player position to collision check

ENDIF

IF collision check detects no collision THEN:

Update player location

ELSE:

Do nothing

ENDIF

Collision Check:

IF player location != wall or enemy locations THEN:

Collision = false

ELSE:

Collision = true

ENDIF

Item Collision Check:

IF player location != item location THEN:

Do nothing

ELSE:

Call item pickup

ENDIF

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

Run feature 1000 each time testing player movement with a collision location and outputting the results to a file.

The results should have the following characteristics:

1. A Boolean displaying if the collision was handled correctly

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

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PWks) | Predecessor Task(s) |
| 1. Requirements Collection | 1 | - |
| 2. Implement Manage Player Controls | 2 | 1 |
| 3. Implement Update Player Position | 2 | 2 |
| 4. Implement Collision Check | 2 | 3 |
| 5. Implement Item Collision Check | 1 | 3 |
| 6. Testing | 2 | 5 |
| 7. Installation | 1 | 6 |

### Pert diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | | 1 | | 6 |
| 5 | | | | |
| 5 | 1 | | 7 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | | 2 | | 9 |
| 6 | | | | |
| 7 | 0 | | 9 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | | 1 | | 10 |
| 7 | | | | |
| 9 | 0 | | 10 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | | 2 | | 5 |
| 3 | | | | |
| 3 | 0 | | 5 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | | 2 | | 3 |
| 2 | | | | |
| 1 | 0 | | 3 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | | 1 | | 1 |
| 1 | | | | |
| 0 | 0 | | 1 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | | 2 | | 7 |
| 4 | | | | |
| 5 | 0 | | 7 | |

### Gantt timeline

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |