Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

## Brief introduction \_\_/3

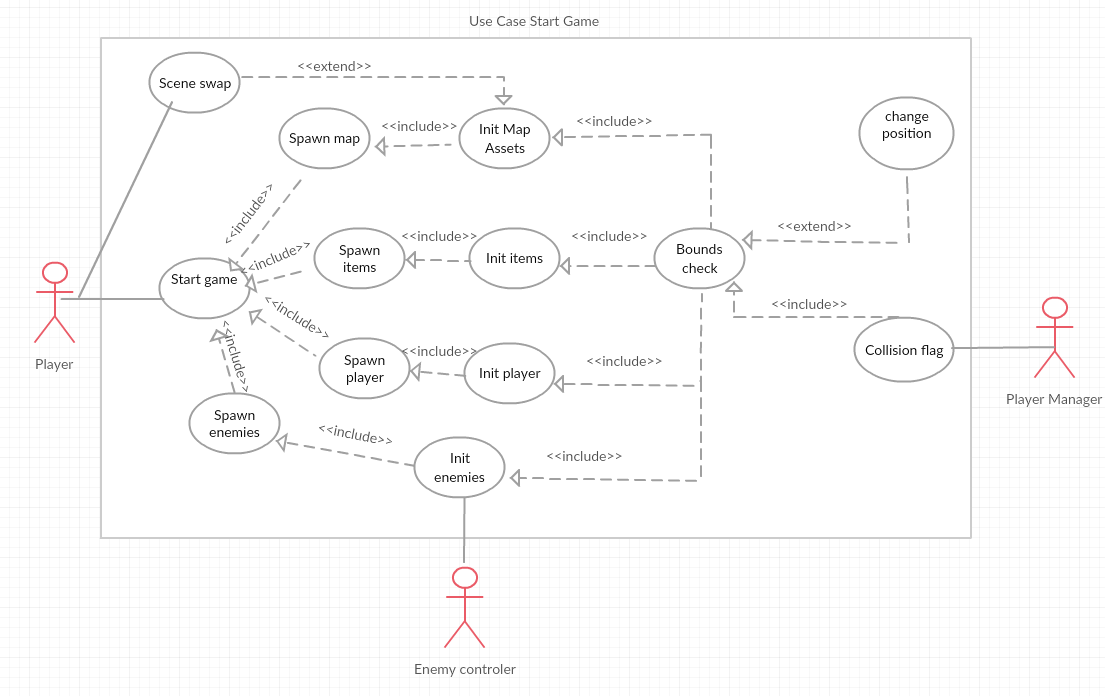
The feature that I will be handling is level design. I manage all asset loading and de-spawning within the world as well as scene management. This includes the player character, enemies, actual objects scenes in the world. I am also responsible on for checking with the position data given by each asset to ensure that collision detection is working properly and none of the assets are spawned outside the world or within the collision boxes and react properly and disallow movement when necessary.

## Use case diagram with scenario \_\_14

[Use the lecture notes in class. Ensure you have at least one exception case]

Example:

### Use Case Diagrams



### Scenarios

**[You will need a scenario for each use case]**

**Name:** Start Game

**Summary:** Initialize all world assets and place them on the map including players, enemies and items.

**Actors:** Player

**Preconditions:** The start button has been pressed

**Basic sequence:**

**Step 1:** Spawn the map at the origin point with initial settings

**Step 2:** Spawn items at location given in initial settings

**Step 3:** spawn enemies at location given in initial settings and enable enemy control.

**Step 4:** Spawn player at location given in initial settings

**Exceptions:**

**Step 1:** [Bounds check] fails on initial map asset locations: reset Asset location to within the bounds of the map.

**Step 2:** [Bounds check] fails on initial item location: reset item location to within the bounds of the map.

**Step 3:** [Bounds check] fails on initial enemy location: reset enemy location to within the bounds of the map.

**Step 4:** [Bounds check] fails on initial player location: reset location to within the bounds of the map.

**Post conditions:** Game starts with all included assets properly placed.

**Priority:** 1

**ID:** C01

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

**Name:** Collision detect

**Summary:** The player character will know that it has collided with a map asset

**Actors:** Player manager

**Preconditions:** The player manager has collided with something

**Basic sequence:**

**Step 1:** [Bounds Check] set collision flag to result

**Exceptions:**

**Step 1:** [spawn player] [init player] has not completed: wait till completion.

**Step 2:** player has collided with something other than a map asset: do not set flag

**Post conditions:** the collision flag is set for a map asset hit.

**Priority:** 1

**ID:** C02

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

**Name:** Enemy spawn

**Summary:** The enemy will spawn into the world and control will be given to the enemy manager

**Preconditions:** The Start game button has been pressed

**Basic sequence:**

**Step 1:** Take initial spawn location data from the enemy manager

**Step2:** [Bounds check] requested location

**Step3:** enemy spawns

**Step4:** control is given to the Enemy Manager

**Exceptions:**

**Step 1:** [Start Game] has not been pressed: wait until completion.

**Step 2:** [bounds check] fails: reset location data to within bounds of the map

**Post conditions:** Enemy spawns in proper location and control is given to the enemy manager.

**Priority:** 2

**ID:** C03

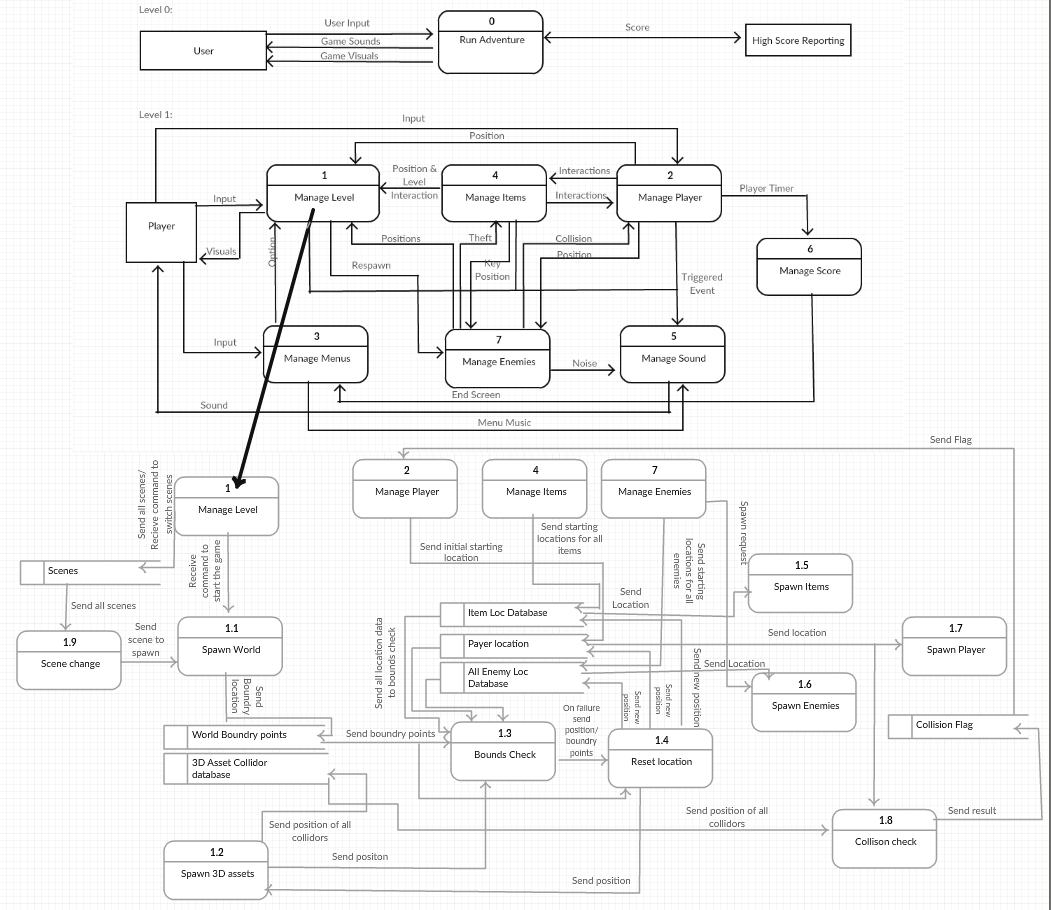
\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## 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]

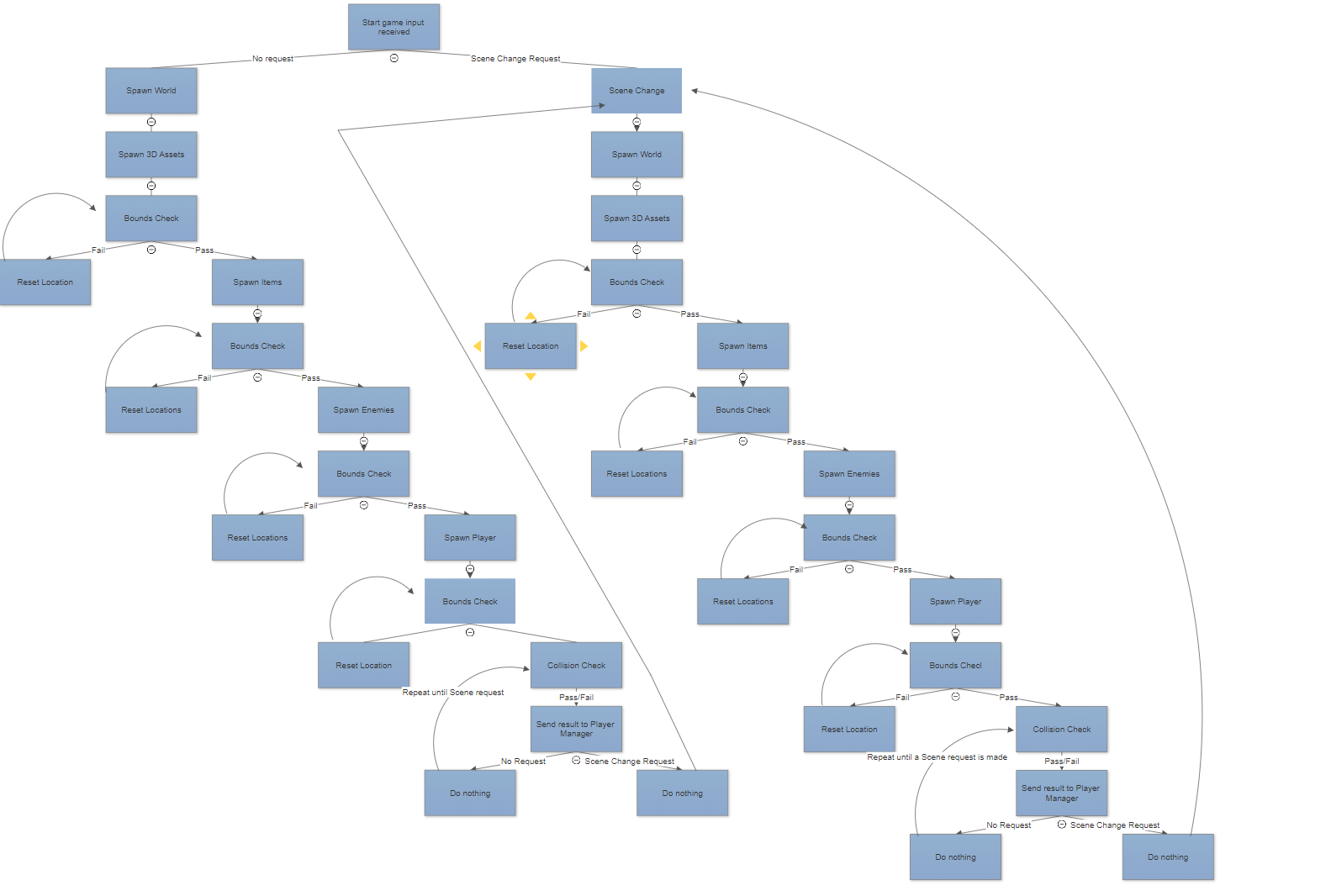
Example:

### Data Flow Diagrams

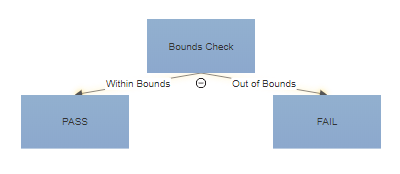


### Process Descriptions

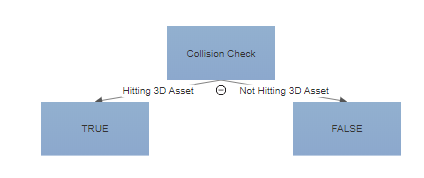
Decision tree for a completed level load



Decision Tree for Bounds Check



Decision Tree for Collision Check



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

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

**Bounds Check Feature**

Run feature 50 times sending output to a file.

The output file will have the following characteristics:

* Number of Fails: 9
* Number of Passes: 0
* Fails will appear no more than 50% of the time
* Passes will appear at least 50% of the time

**Example for Bounds Check feature**

|  |  |  |  |
| --- | --- | --- | --- |
| Output | 3D Points(X,Y,Z) Check | 3D Point origin Box(X,Y,Z)(X,Y,Z) | Notes |
| Pass | (0,0,0) | (0,0,0)(100,20,0) | This passes because it is not outside the bounds |
| Pass | (15,-3,20) | (0,-300,20)(100,0,20) | This passes because it is still within the bounds |
| Fail | (0,9,20) | (0,10,20)(100,10,20) | This will fail because it is below the bounds by 1 unit |
| Fail | (0,100,40) | (0,100,45)(100,101,45) | This will fail because it is 5 units below bounds |

**Collision Check Feature**

Run feature 50 times sending output to console.

The console will have the following characteristics:

* Will always print true when hitting an object tagged as a 3D Asset
* Will always print False when not in contact.

**Example for Bounds Check feature**

|  |  |  |  |
| --- | --- | --- | --- |
| Output | Player Location (X,Y,Z) | 3D Asset Collider box location (X,Y,Z)(X,Y,Z) | Notes |
| True | (0,0,0) | (0,0,0)-(0,10,20) | True for it is within its box |
| True | (15,3,20) | (0,300,20)-(100,0,20) | This passes because it is still touching the box |
| False | (0,9,20) | (0,10,20)-(100,10,20) | This will fail because it is not touching a collider |
| False | (0,150,40) | (0,100,45)-(100,101,45) | This will fail because it is not touching a collider |

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

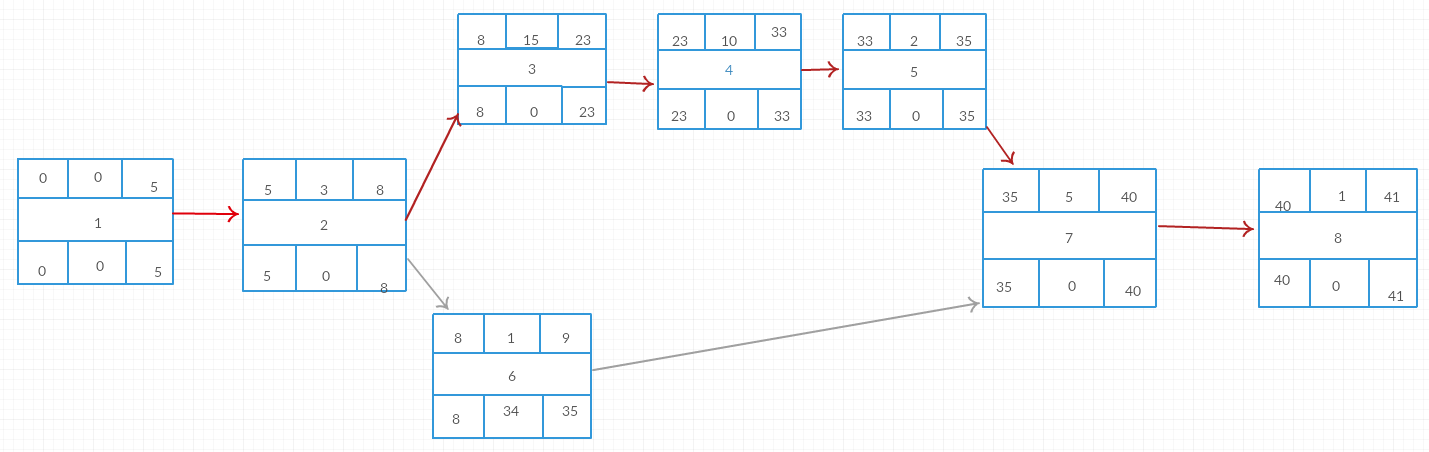
[Figure out the tasks required to complete your feature]

Example:

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PHrs) | Predecessor Task(s) |
| 1. Obtain 3D Assets | 5 | - |
| 2. Implement World Loading | 3 | 1 |
| 3. Implement Bounds Check | 15 | 2 |
| 4. Implement Item/Player/Enemy Loading | 10 | 2, 3 |
| 5. Implement Player Manager Collision Check | 2 | 4 |
| 6. Implement Scene Swapper | 1 | 2 |
| 7. Testing | 5 | 1,2,3,4,5,6 |
| 8. Installation | 1 | 7 |

### Pert diagram



### Gantt timeline

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  | 2,3 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |
| 6 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1,2,3, | 4,5,6 |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |  |
|  | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 |