Name\_\_\_\_Gigi Young\_\_\_\_\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

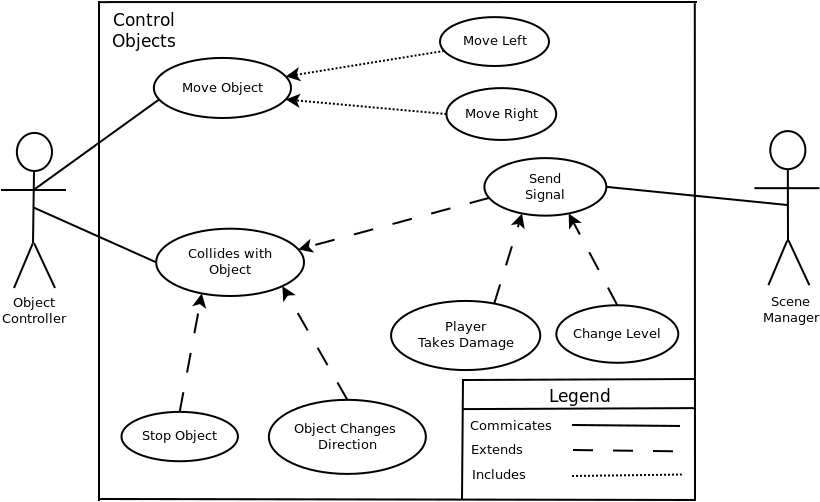
[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

## Brief introduction \_\_/3

I am in charge of instantiating the objects for our game, Fish N’ Chips. An object is anything that is able to interact with another object to produce a tangible effect, excluding the main character. In Fish N’ Chips, enemies, collectibles, and platforms are all objects that can also interact with the main character.

## Use case diagram with scenario \_\_14

### Use Case Diagram



### Scenarios

**Name:** Move Object

**Summary:** The object controller moves an instance of an object that can move (shark or banana).

**Actors:** Object Controller

**Preconditions:** The game has started and scene is loaded.

**Basic sequence:**

**Step 1:** An instance of a moving object is instantiated.

**Step 2:** The object’s position is updated via time slot.

**Step 3:** The position of the object is updated via the Object Controller’s algorithm for that specific object.

**Exceptions:**

**Step 1:** An object (shark) is moving.

**Step 2:** A banana is thrown by the main character and collides with the shark.

**Step 3:** The shark is stunned and rendered unable to move for a duration

**Post conditions:** The object occupies a different position than before.

**Priority:** 2

**ID:** C01

**Name:** Move Left

**Summary:** The object controller moves an instance of an object that can move (shark or banana) left.

**Actors:** Object Controller., Scene Manager

**Preconditions:** The game has started and scene is loaded.

**Basic sequence:**

**Step 1:** An instance of a moving object is instantiated.

**Step 2:** The object’s position is updated via time slot to move leftwards.

**Step 3:** The position of the object is updated to the left of where it was previously via the Object Controller’s algorithm for that specific object.

**Exceptions:**

**Step 1:** An object (shark) is moving towards the right.

**Step 2:** A banana is thrown by the main character and collides with the shark.

**Step 3:** The shark is stunned and rendered unable to move for a duration

**Post conditions:** The object occupies a position left of where it was previously

**Priority:** 2

**ID:** C02

**Name:** Move Right

**Summary:** The object controller moves an instance of an object that can move (shark or banana) right.

**Actors:** Object Controller., Scene Manager

**Preconditions:** The game has started and scene is loaded.

**Basic sequence:**

**Step 1:** An instance of a moving object is instantiated.

**Step 2:** The object’s position is updated via time slot to move rightwards.

**Step 3:** The position of the object is updated to the right of where it was previously via the Object Controller’s algorithm for that specific object.

**Exceptions:**

**Step 1:** An object (shark) is moving towards the right.

**Step 2:** A banana is thrown by the main character and collides with the shark.

**Step 3:** The shark is stunned and rendered unable to move for a duration

**Post conditions:** The object occupies a position right of where it was previously

**Priority:** 2

**ID:** C03

**Name:** Collides with Object

**Summary:** An instance of an object in the game collides with an instance of a different object.

**Actors:** Object Manager, Scene Manager (extension cases.

**Preconditions:** The two objects that will be colliding are loaded into the scene.

**Basic sequence:**

**Step 1:** One or both objects are moving.

**Step 2:** The hitboxes (the area contained by an object) of the two objects make contact

**Step 3:** The objects realized that contact with another object has been made

**Step 3:** An effect is incurred depending on the type of object collision

**Exceptions:**

**Step 1:** The two objects that collide do not incur a side effect

**Step 2:** No nothing happens

**Post conditions:** Depending on the type of object collision, it is either resolved by the Object Manager or a signal is sent to the Scene Manager. Some effects include object movement cesation, player health reduction, player health increase, loading the next level, etc.

**Priority:** 2

**ID:** C04

**Name:** Stop Object

**Summary:** An instance of an object in the game collides with an instance of a different object causes one of the previously moving objects to stop moving.

**Actors:** Object Manager

**Preconditions:** The two objects that will be colliding are loaded into the scene.

**Basic sequence:**

**Step 1:** One or both objects are moving.

**Step 2:** The hitboxes (the area contained by an object) of the two objects make contact

**Step 3:** The objects realized that contact with another object has been made

**Step 4:** The moving object stops moving because it has collided with an object that is designed to stop moving (a platform for example).

**Exceptions:**

**Step 1:** Objects do not collide

**Step 2:** Nothing happens

**Post conditions:** A moving object’s movement is stopped as the result of colliding with a platform.

**Priority:** 2

**ID:** C05

**Name:** Object Changes Direction

**Summary:** An instance of an object in the game collides with an instance of a different object causes one of the moving objects to change movement direction (e.g. left to right).

**Actors:** Object Manager

**Preconditions:** The two objects that will be colliding are loaded into the scene.

**Basic sequence:**

**Step 1:** One or both objects are moving.

**Step 2:** The hitboxes (the area contained by an object) of the two objects make contact

**Step 3:** The objects realized that contact has been made

**Step 4:** The moving object starts moving in the opposite direction, e.g. a shark collides with a platform and start moving in the opposite direction.

**Exceptions:**

**Step 1:** Objects do not collide

**Step 2:** Nothing happens

**Post conditions:** A moving object’s movement reversed from colliding with a specific object (platform).

**Priority:** 2

**ID:** C06

**Name:** Send Signal

**Summary:** An instance of an object in the game collides with an instance of a different object causes one of the moving objects to change movement direction (e.g. left to right).

**Actors:** Scene Manager

**Preconditions:** Two particular objects have collided that would result in a need for the a change to be made by the Scene Manager.

**Basic sequence:**

**Step 1:** The signal is sent from one of the collided objects to the Scene Manager.

**Step 2:** The Scene Manager enacts the change specific to that particular signal

**Post conditions:** A scene changing effect occurred, e.g. level change, player health reduction.

**Priority:** 2

**ID:** C07

**Name:** Player Takes Damage

**Summary:** A shark or steam collides with the main character, causing player health to decrease.

**Actors:** Scene Manager

**Preconditions:** There are instances of sharks and or steam in the current game level.

**Basic sequence:**

**Step 1:** The main character collides with a shark or steam.

**Step 2:** The shark or steam object sends a signal to the Scene Manager.

**Step 3:** The Scene Manager receives this particular signal and reduces the player’s health as a result.

**Post conditions:** The player’s health is lower than before.

**Priority:** 2

**ID:** C08

**Name:** Change Levels

**Summary:** The player collides with the door/exit object, which results in the next level being loaded.

**Preconditions:** The must exist an instance of a door/exit in the current level

**Basic sequence:**

**Step 1:** The main character collides with the door/exit.

**Step 2:** The door/exit object sends a signal to the Scene Manager.

Step 3: The Scene Manager receives the signal, loads a new level and deallocates the current level.

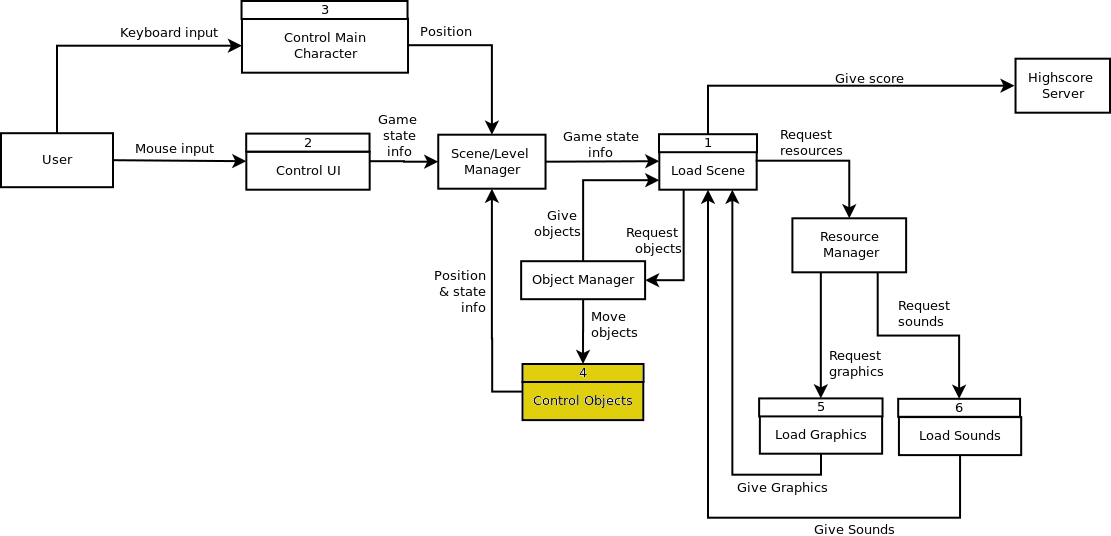
**Post conditions:** A new level within the scene is loaded.

**Priority:** 2

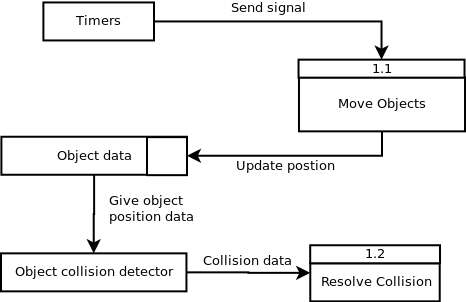
**ID:** C09

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

Diagram 0



Control Objects



### Process Descriptions

Move Objects:

IF signal from timer THEN

IF shark THEN

move by current velocity

IF banana AND is being thrown THEN

move in direction of throw by current velocity

Resolve Collision:

IF collision of (shark and player) OR (steam and player) THEN

signal scene to reduce player health

IF collision of banana(stationary) and player THEN

signal scene to give player a banana

IF collision of door/exit with player THEN

signal scene to load next level

IF collision of anything with platform THEN

stop object that collided with platform

IF collision of thrown banana and shark THEN

stop shark from moving for a set period of time

//possible extra features if enough time:

IF collision of shark and steam THEN

send shark signal “to be cooked”

shark object changes to cooked shark object

IF collision of cooked shark and player THEN

send signal to scene to increase player health

IF collision of player with oxygen tank THEN

send signal to scene to increase oxygen level

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

Note: Run each test 5 times or until expect outputs occur

Note: Each bullet point will be a separate sub-test

**Test for moving objects**

Create a QT scene and

* Call 10 instances of sharks to be spawned randomly within the scene
* Call 10 instances of thrown bananas to be spawned randomly within the scene

Input: Objects (sharks and thrown bananas)

Output: Observe object movement

**Test for platform collision**

Create a QT scene and add platforms to the edges of the scene to create a closed box and

* Call 10 instances of sharks to be spawned randomly within the scene
* Call 10 instances of thrown bananas to be spawned randomly within the scene

Input: Objects (platforms, sharks, and thrown bananas)

Output: Observing object collision behavior with the platforms

**Test for player collision with objects**

Create a QT scene with platforms on the edges to form a closed space and spawn an object of each type (excluding more platforms) in a specific sectors in order to be able to differentiate object types ( assuming graphics have not yet been implemented) and

* Collide the player with each object in turn

Input: All objects and main character with keyboard input

Output: Observing object collision of main character with each object

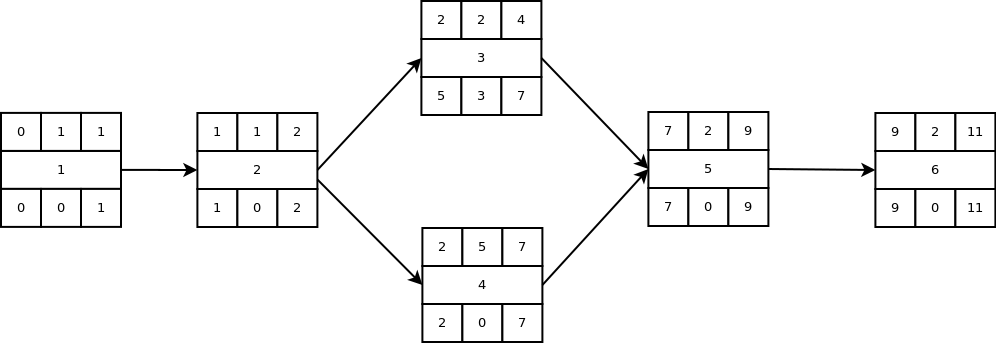
For expected outputs, refer to process descriptions for Diagram 1

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

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Wks) | Predecessor Task(s) |
| 1. Object Design | 1 | - |
| 2. Object Interface Design | 1 | 1 |
| 3. Interface Documentation | 2 | 2 |
| 4. Programming | 5 | 2 |
| 5. Testing | 2 | 3,4 |
| 6. System Implementation | 2 | 5 |

### Pert diagram



### Gantt Chart

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  | 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  | 3 |  |  |  |  |  |  |  |  |
| 4 |  |  | 4 |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  | 5 |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  | 6 |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |