Infosys Ltd.

Interactive 3D Game Development



|  |
| --- |
| **Utkarsh Sinha(523384), Varun Agrawal(523385),**  **Priyanka Anand(523386), Mousumi Kundu(523387)** |
| **Date: 10th May, 2015**  **Reviewer : Mr. Rahul Roychowdhury** |

**Introduction**

Unity 3D is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices. An attempt to bring platform games into 3D. It brings an enjoyable experience to the game lovers. The implementation of the techniques is illustrated through the PC Game and described in detail with the complete code and Gaming Interface. Therefore, this project results in a successful definition of the instructions for the development techniques and a functional PC Based Gaming Portal Application.

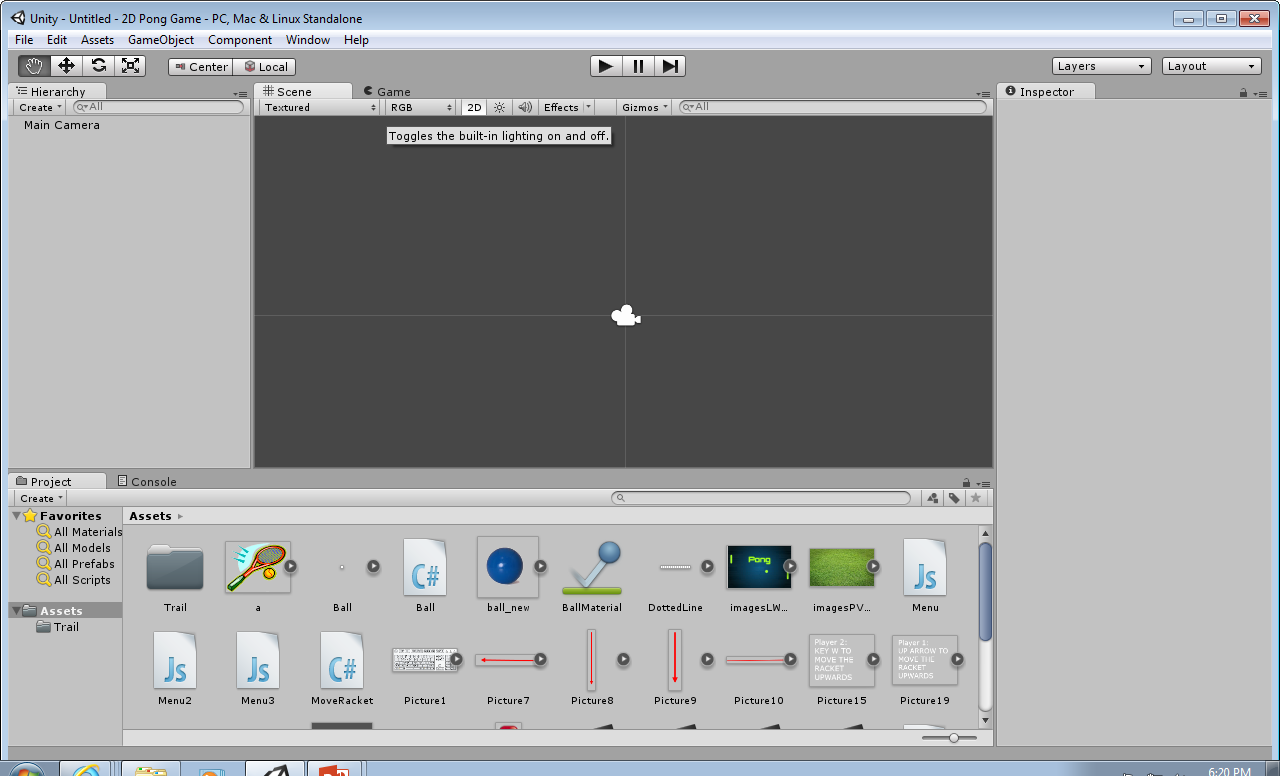
* The aim of our project is to develop a gaming portal having different simple 2D interactive games.
* These games are developed using Unity 3D.
* It is a flexible and powerful development platform for creating multiplatform 3D and 2D games and interactive experiences.

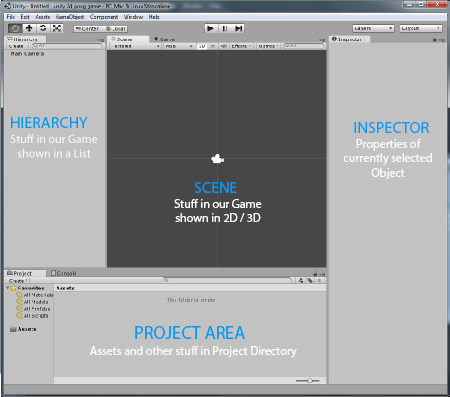
**Unity Basics**

The workflow in Unity goes something like this: create an entity to serve a role in the game (blank Game Objects can be used for abstract logical tasks). Then, either write or find a class file, and add it to the entity as a script (using the ‘add component’ button in the ‘inspector’ view. Then run, test, debug, repeat until it works and move on to the next element of the game.

Unity comes with a number of basic view tabs that can be laid out in various ways to the taste of the user. The big five are the ‘game’ tab, the ‘scene’ tab, the ‘inspector’ tab, the ‘project’ tab, and the hierarchy tab. The game tab, when the ‘play’ button is depressed, displays a running instance of the game that the user can interact with and test. The ‘scene’ tab provides a static, editable version of the game world. The ‘inspector’ tab allows the user to modify individual entities in the game world by selecting them in the ‘editor’ tab. The ‘project’ tab allows the user to browse through the project’s files and drag models, materials, and other resources into the ‘editor’ tab to place them in the game world. Finally, the ‘hierarchy’ tab shows all objects in the world, allowing you to find distant objects in the scene, and parent entities to one another by clicking and dragging.

**Unity Workspace**

****

**Unity Elements**

The four elements are:

* **Hierarchy:**

On the left side we can see the Hierarchy, which contains a list of all the things that are currently in our game. As we can see, currently it's only the Main Camera, which was added by Unity automatically.

* **Project:**

The Project Area contains all kinds of Assets like Textures, 3D Models or Scripts. If we want, we can use those Assets in our game by dragging them into the Hierarchy or into the Scene.

* **Scene:**

We can see the game world in the Scene view. We can find any game object in the Scene by double clicking it in the Hierarchy. To navigate around the Scene, we just click into it once and then use the left, right and middle mouse buttons.

* **Inspector:**

The Inspector shows the properties of the currently selected object(s). For example, if we select the Main Camera in the Hierarchy then we can see its Position, Rotation, Name and all other things that are relevant about it.

**Scene controls**

At the top-left of your screen, you'll see four controls that help you move around your **Scene**,

and position **Game Objects** within it. These controls are mapped to the *Q*, *W*, *E*, and *R* keys

on your keyboard. From left to right, they are:



**Hand tool (Q)**: Use it to click-and-drag around your scene. Hold down the *Alt* key on your keyboard to rotate the view. Hold down the *Ctrl* key (Windows) or the *Command* key (Apple) to zoom in and out. Your mouse wheel will also zoom the scene. Hold down the *Shift* key to pan, zoom, and rotate in larger increments to speed things up.

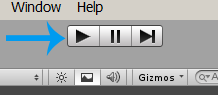
**Move tool (W)**: This tool lets you move the **Game Objects** around your scene. You can either drag the object(s) around by the X, or Y, or Z-axis handles, or by the square in the center for freeform movement. Holding down the *Ctrl* key will snap movement to set grid increments.

**Rotate tool (E)**: Use it to spin your objects around using a neat spherical gizmo. The red, green, and blue lines map to the X, Y, and Z axes.

**Scale tool (R)**: This tool works much the same as the **Move** and **Rotate** tools. Use it to make your **Game Objects** larger or smaller. Dragging an X, Y, or Z handle will non-uniformly scale (squash and stretch) the object, while dragging the gray cube in the center will uniformly scale it.

**Playing:**

* On pressing Play button, we see the Game World through the eyes of Camera.
* To go back to the editing mode, press the Play button again (to stop playing).



**Coding in Unity**

* *using UnityEngine; -* This line tells C# that we want to use Unity’s libraries, which allow us to connect to the Unity game engine.
* *public class Mook :* [*MonoBehaviour*](http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.html) *-* This line actually declared the class and its name.
* *private float health; -*This declares a private class variable (which can only be changed from inside the class). The variable is given a value in Start().
* *void Start () -* This declares a method called ‘Start.’ Start is a special method that runs only once, when the game initially launches.
* void Update() - Update is another special method, which runs on every frame. Most of your game logic comes here.
* *Note -// if you encounter the player on the road, kill him -*This line is a comment (any line starting with a double slash is ignored by C#). Comments are used to remind yourself of what particular bits of code do.

Once you have a script attached to an object, you can revise it by double clicking on it in the ‘inspector.’ This opens MonoDevelop, the default development environment for Unity. In essence, Monodevelop is a text editor with features specifically optimized toward programming. Keywords and comments are highlighted in blue and green, and numerical values and strings appear in red. If you’ve used Eclipse or other IDE’s, MonoDevelop is very similar. You can ‘build’ your scripts from inside the editor, to check for syntax errors, like so:

In general, to get your script to interact with Unity, you’ll need to reference elements that the object holding the script possesses. You can then call methods or set variables for each of these elements to enact the changes you want.

If you want a script on an object to affect the properties of a different object, you can create an empty GameObject variable in your script, and use the inspector to assign it to another object in the scene. A list of the elements an object might have is as follows:

* Transform
* Cube
* Box Collider
* Mesh Renderer

Each of these aspects of the object can be influenced from within a script.

**Unity Entities**

* **GUI Elements**

Traditional GUI sprites and text can be displayed using the ‘GUI Text’ and the ‘GUI Texture’ Game Objects in the editor. However, a more robust and realistic way to handle UI elements is to use the 3D text to place elements into the game world as entities. In the ‘hierarchy’ view, these gameplay elements can be dragged onto the main camera to make them children, ensuring that they move and rotate with the camera.

GUI elements (text and textures) can have their size and scale adjusted using the relevant fields in the inspector tab.

* **Materials**

Materials are combinations of textures and shaders, and can be dragged directly onto game objects from the project tab. To import a texture, just convert it to a jpg, png, or bmp, and drag it into the ‘assets’ folder under the Unity project directory (which appears in ‘My Documents’ by default). After a few seconds, a loading bar will appear in the editor. When it finishes, you’ll be able to find the image as a texture under the ‘project’ tab.

* **Meshes**

Meshes are the way 3D geometry is represented in Unity. The user can either use Unity’s built-in ‘primitive’ objects (cubes, spheres, cylinders, etc).Unity supports a variety of 3D formats, including .3ds.

The basic tools for manipulating meshes are the scaling, rotation, and translation buttons in the upper left corner of the interface. These buttons add control icons to the models in the editor view, which can then be used to manipulate them in space. To alter the texture or physics properties of an object, select them and use the ‘inspector’ view to analyze the ‘material’ and ‘rigidbody’ elements.

* **Lights**

Lights are GameObjects which project radiance onto the world. If there are no lights in your scene, all polygons are drawn at the same brightness level, giving the world a ‘flat’ look.

Lights can be positioned, rotated, and have several internal characteristics that you can customize. The ‘intensity’ slider controls the brightness of the light, and the ‘range’ controls how quickly it fades out.

The three main kinds of light are ‘spot’, ‘point’, and ‘directional.’ Spot lights have a location in 3D space and project light only in one direction in a cone of variable angle. Spot lights can cast shadows. Point lights have a location in 3D space, and cast light evenly in all directions. Point lights do not cast shadows. Directional lights, finally, are used to simulate sunlight: they project light in a direction as though from infinitely far away. Directional lights affect every object in the scene, and can produce shadows.

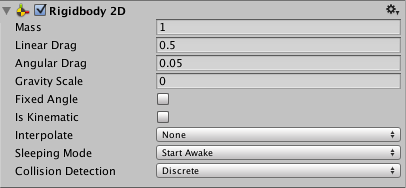
* **Particle Systems**

Particle systems are the term for Unity GameObjects that generate and control hundreds or thousands of particles simultaneously. Particles are small, optimized 2D objects displayed in 3D space. Particle systems use simplified rendering and physics, but can display thousands of entities in real time without stuttering, making them ideal for smoke, fire, rain, sparks, magic effects, and more.

There are a lot of parameters that you can tweak to achieve these effects, and you can access them by spawning a particle system under the component editor, selecting the particle system, and then opening the ‘inspector’ tab. You can change the size, speed, direction, rotation, color, and texture of each particle, and set most of those parameters to change over time as well. Under the ‘collision’ attribute, if you enable it and set the simulation space to ‘world’ you’ll get particles that will collide with objects in the world, which can be used for a number of realistic particle effects, including rain, moving water, and sparks.

**2D Physics Reference**

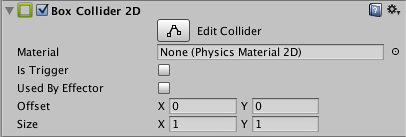
* **Rigidbody 2D**

Rigidbody 2D component places an object under the control of the physics engine. Many concepts familiar from the standard Rigidbody component carry over to Rigidbody 2D, with the difference that in 2D, objects can only move in the XY plane and can only rotate on an axis perpendicular to that plane.

| **Property:** | **Function:** |
| --- | --- |
| Mass | Mass of the rigidbody. |
| Linear Drag | Drag coefficient affecting positional movement. |
| Angular Drag | Drag coefficient affecting rotational movement. |
| Gravity Scale | Degree to which the object is affected by gravity. |
| Fixed Angle | Angle according to which the rigidbody rotates when forces are applied. |
| Is Kinematic | The rigidbody is moved by forces and collisions or not. |
| Collision Detection | How collisions with other objects are detected. |
| Discrete | A collision is registered only if the object’s collider is in contact with another during a physics update. |
| Continuous | A collision is registered if the object’s collider appears to have contacted another between updates. |

* **Box Collider 2D**

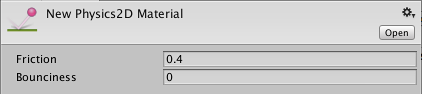
The Box Collider 2D component is a collider for use with 2D physics. The collider’s shape is a rectangle with a given position, width and height in the local coordinate space of a Sprite. Note that the rectangle is axis-aligned, ie, its edges are parallel to the X or Y axes of local space.



| **Property:** | **Function:** |
| --- | --- |
| Material | A physics material that determines properties of collisions, such as friction and bounce. |
| Is Trigger | The collider behave as a trigger or not. |
| Offset | The local offset of the collider geometry. |
| Size | The size of the box in local space units. |

* **Physics Material 2D**

A Physics Material 2D is used to adjust the friction and bounce that occurs between 2D physics objects when they collide. You can create a Physics Material 2D from the Assets menu (Assets > Create > Physics2D Material ).



| **Property:** | **Function:** |
| --- | --- |
| Friction | Coefficient of friction for this collider. |
| Bounciness | The degree to which collisions rebound from the surface. A value of 0 indicates no bounce while a value of 1 indicates a perfect bounce with no loss of energy. |

**Transform**

The transform functions for a GameObject in Unity control the physical parameters of that object: its scale, its position, and its orientation. Example:

**transform.position = newPositionVector3;**

**Renderer**

The renderer functions in Unity allow you to control the way the surfaces of props are rendered on-screen. You can reassign the texture, change the color, and change the shader and visibility of the object.

**Input**

A game that doesn’t take any input from the user isn’t much of a game. There are a lot of different kinds of input you can read in, and almost all of them are accessible through the Input and KeyCode objects. Example:

**Vector3 mousePos = Input.mousePosition;**

### **Collision**

Often, when building a game, you’d like a collision to result in some change-of-state in your code, beyond just physics simulation. For this, you’ll need a collision detection method.

**void OnCollisionEnter(Collision other){**

**//do things here**

**}**

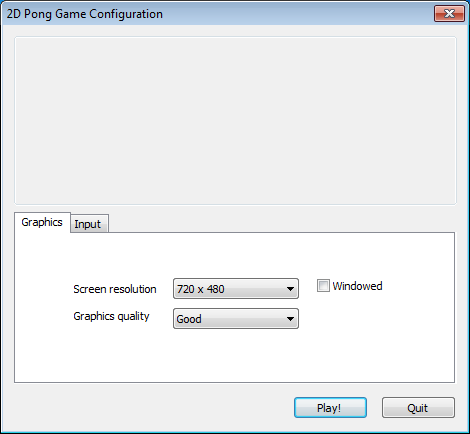
This method will automatically run during the first frame that another object touches your object.

**Pong Game**

* One of the games of the portal is a multimedia based 2D game named “PONG GAME”.
* It is a video game developed in Unity 3D.
* It is a two player game where each player has a racket and they need to save the ball from touching the walls on their side.
* When the ball touches any collider, a plop sound can be heard.

**Pong Game Configuration**

* This is the first screen which is visible on double clicking the game icon.
* Here we can change the graphics quality as well as screen resolution.



**Features of the Game:**

* The ball has a trail effect.
* The score board for individual players is maintained.
* The grass background makes the game close to reality.
* There is plop sound whenever the ball touches a collider.
* User friendly menu and instructions screen.

**Main Menu**

This is the menu of our game which has 3 buttons:

* Play - The play button takes us to the game window.
* Instructions - Tells us how to play the game and about player controls.
* Quit - Quits the running application.

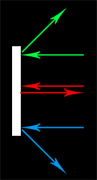
**Instructions Page**

* This shows all the controls as well as it has a go back button which takes us back to the menu page.
* Player 1 uses Up and Down Arrow Keys and Player 2 uses W and S keys to move their rackets respectively.

****

**Description**

* This particular scene consists of a ball and 2 rackets for 2 players, along with a Restart and Quit button.
* The right player scores a point if the ball hits the left wall. The left player scores a point if the ball hits the right wall. If it hits the top or bottom wall then it bounces off. Each player will have a racket that can be moved up and down to hit back the ball.
* The scoreboard keeps updating the score which helps us to decide the winner at the end.
* There is no fixed end point and one can continue playing the game as long as he/she wishes to.
* To end the game Quit button needs to be clicked.

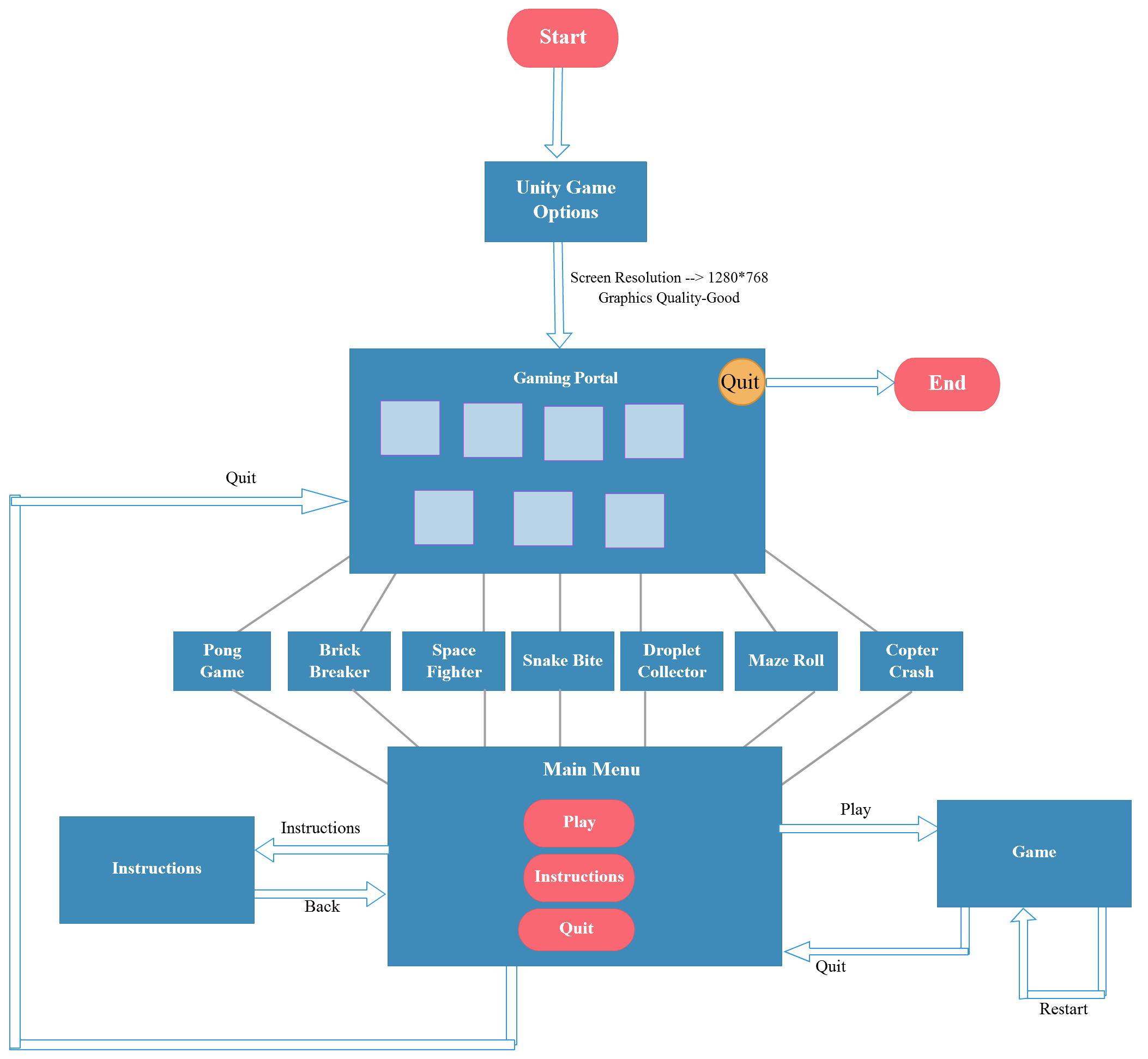
**Deflection of Ball**

Our rackets should have an influence on the ball's outgoing angle:

* If the racket hits the ball at the top corner, then it should bounce off towards our top border.
* If the racket hits the ball at the center, then it should bounce off towards the right, and not up or down at all.
* If the racket hits the ball at the bottom corner, then it should bounce off towards our bottom border.

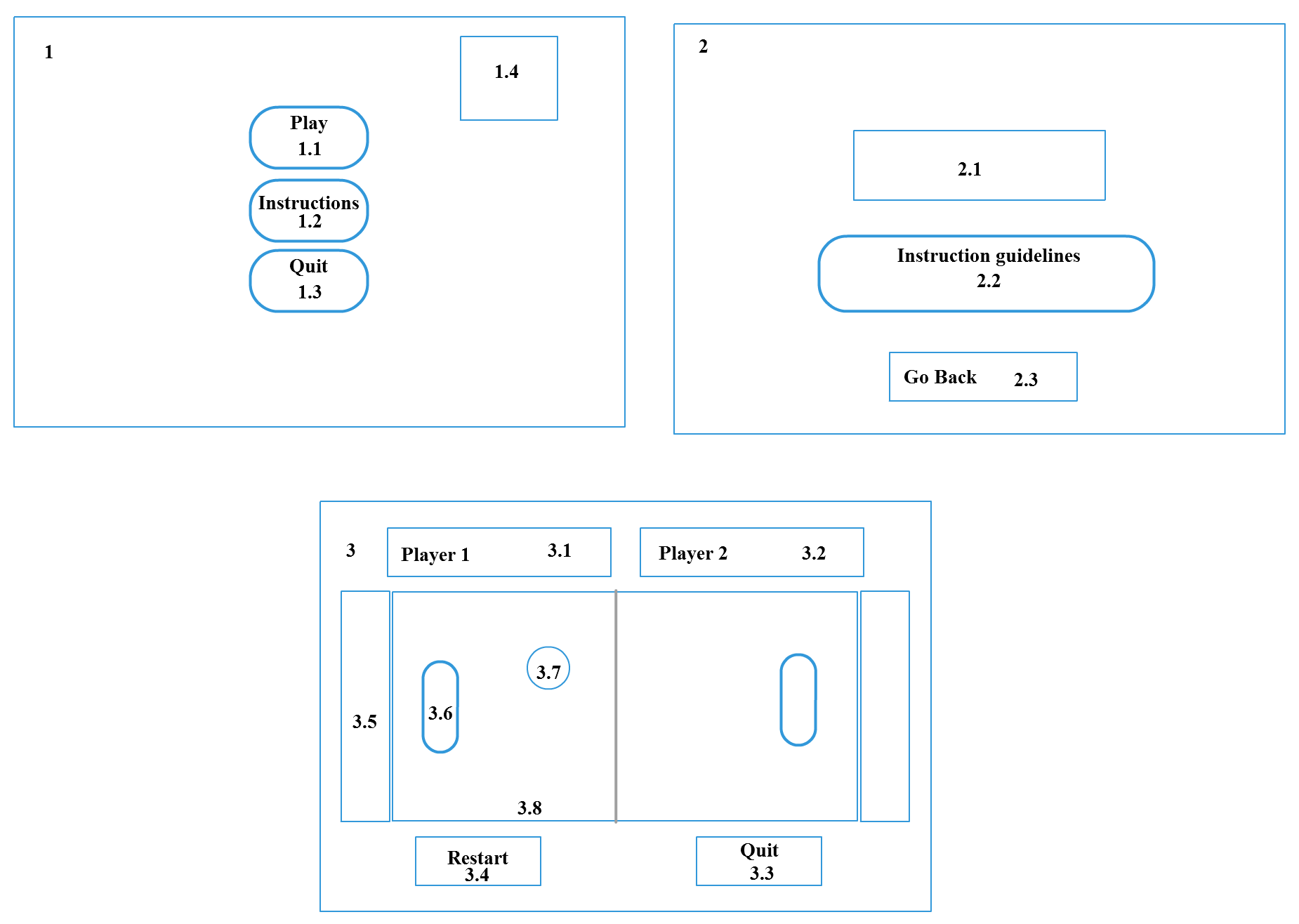
**Sequence Flowchart Diagram**

* The below diagram depicts the flow of different screens in Gaming Portal.
* When we start the game the Unity Game configuration window pop ups from which we can choose the screen resolution and windowed/non windowed mode.
* After that we can see the Gaming Portal screen which has seven games and a close button which quits the application.
* Each of the seven games has three options Play, Instructions and Quit which takes us to their respective screens.
* On clicking the play button we can start playing our game or we can also quit the running application.
* On clicking the instructions button we move to the instructions page that tells us about the controls of the game. It also has a back button which brings us back to the main menu.
* Each game has restart button which loads the game from the beginning.
* We have the quit button which brings us back to the Gaming Portal.



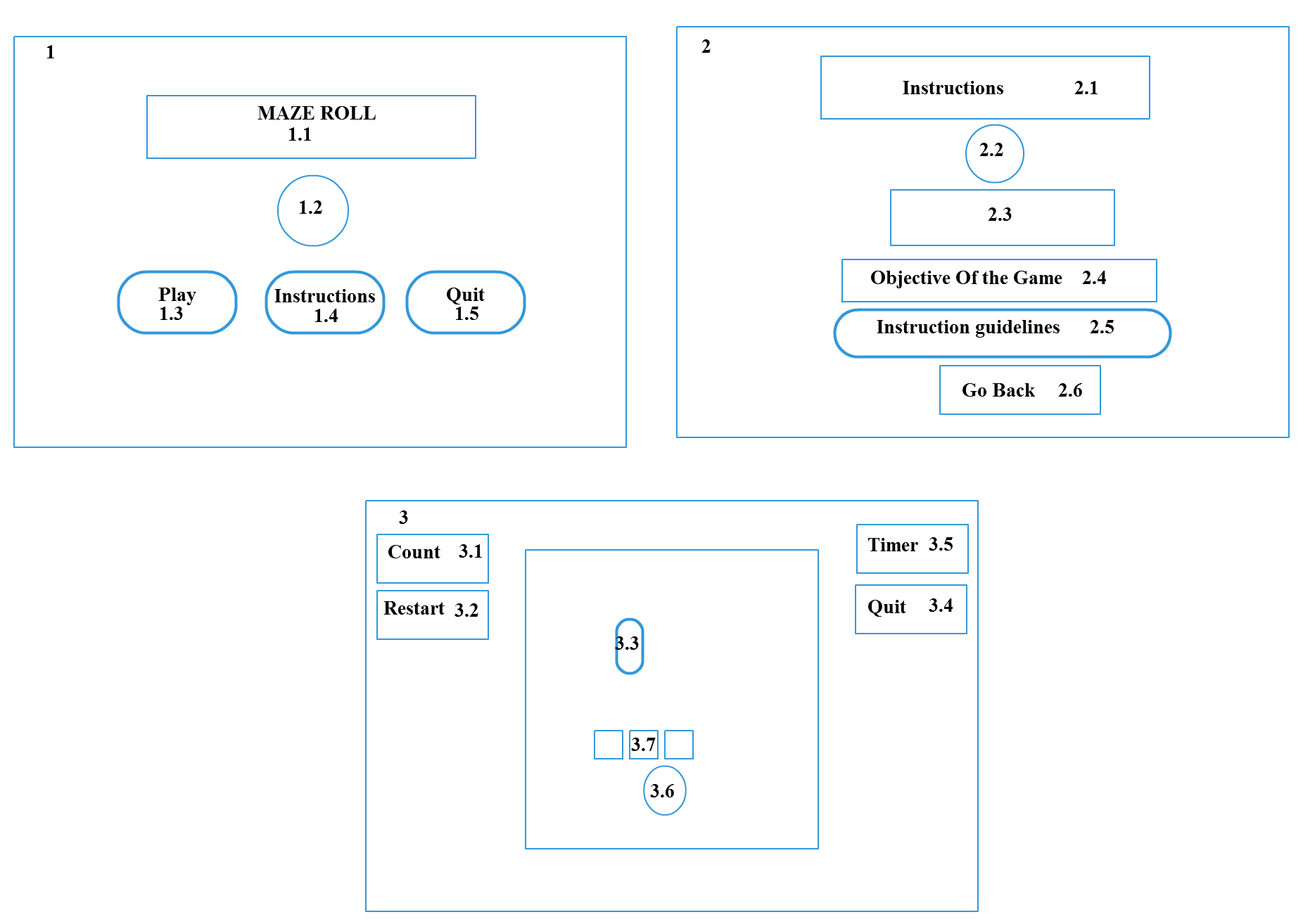
**Story Board Diagrams**

* **Pong Game**

****

* 1. The main menu screen is a Pong image of type JPEG.
  2. Play Button which is a GUI Button.
  3. Instructions Button which is a GUI Button.
  4. Quit Button which is a GUI Button.
  5. Racket logo which is a PNG image.
  6. The instructions screen is an image of type JPEG.
  7. Keyboard image with written controls which are PNG images.
  8. Instruction Guidelines which is a GUI Text.
  9. Back Button which is a GUI Button type.
  10. The game screen is a grass area image of type JPEG.

3.1 Player 1 Score which is a GUI Text.

* 1. Player 2 Score which is a GUI Text.
  2. Restart Button which is a GUI Button.
  3. Quit Button which is a GUI Button.
  4. Audience picture which is a PNG image.
  5. Racket which has Sprite Renderer, Box Collider 2D, Rigid Body 2D and appropriate script attached to it.
  6. Ball which has Sprite Renderer, Box Collider 2D, Rigid Body 2D, appropriate script, Tail Renderer and Audio Source attached to it.
  7. Wall which is the boundary and has Sprite Renderer and Box Collider 2D.
* **Maze Roll**

**3.8**

1. The main menu screen is a Maze image of type JPEG.
   1. Maze Roll heading which is a GUI Text.
   2. Ball which is a texture and has rotating script attached.
   3. Play Button which is a GUI Button.
   4. Instructions Button which is a GUI Button.
   5. Quit Button which is a GUI Button.
2. The instructions screen is an image of type JPEG.

2.1 Instructions heading which is a GUI Text.

2.2 Ball which is a texture and has moving script attached.

2.3 Keyboard image with arrows which are PNG images.

* 1. Objective which is a GUI Text.
  2. Instruction Guidelines which is a GUI Text.
  3. Back Button which is a GUI Button.

1. The game screen is a 3D Maze image of type JPEG.
   1. Count which is a GUI Text keeps track of items collected.

3.2 Restart Button which is a GUI Button.

3.3 Capsules which have Box Collider and Mesh Renderer attached.

3.4 Quit Button which is a GUI Button.

3.5 Timer which is a GUI Text keeps track of time remaining.

3.6 Ball which has Sphere Collider, Mesh Renderer, Rigidbody and appropriate

script attached.

3.7 Boundary blocks which have Mesh Renderer and Box Collider.

3.8 Ground which has Mesh Renderer and Mesh Collider.

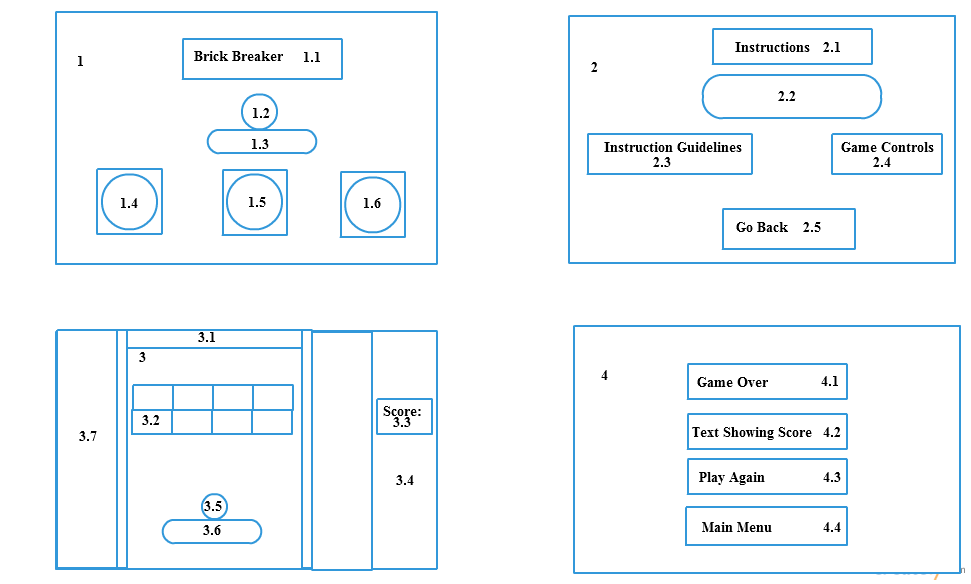
* **Brick Breaker**

****

3.9

3.8

3.3.292185185188

****

1.The main menu screen is an image of type JPEG.

1.1 Brick Breaker which is a GUI Text.

1.2 Ball which has Sprite Renderer, Box Collider 2D, Rigid Body 2D, Ball script

attached.

1.3 Racket which has Sprite Renderer, Box Collider 2D, Rigid Body 2D.

1.4 Play Button which is a GUI Button.

1.5 Instructions Button which is a GUI Button.

1.6 Quit Button which is a GUI Button.

2. The instructions screen is an image of type PNG.

2.1 Instructions which is a GUI Text.

2.2 Keyboard image which is an image of type GIF.

2.3 Instructions guidelines text which is an image of type PNG.

* 1. Game Controls which is a GUI Text.

2.5 Go Back Button which is a GUI Button.

3. The game screen is an image of type PNG.

3.1 Borders which have Sprite Renderer and Box Collider 2D.

3.2 Bricks which are images of type PNG and have Sprite Renderer, Box

Collider 2D and appropriate scripts attached.

3.3 Score which is a GUI Text.

3.4 Score Board which is an image of type JPEG.

3.5 Ball which has Sprite Renderer, Box Collider 2D, Rigid Body 2D and

appropriate scripts attached.

3.6 Racket which has Sprite Renderer, Box Collider 2D, Rigid Body 2D and

appropriate scripts attached.

* 1. Audience which is an image of type JPEG.
  2. Retry Button which is a GUI Button.

3.9 Main Menu Button which is a GUI Button.

4. The game over screen is an image of type PNG.

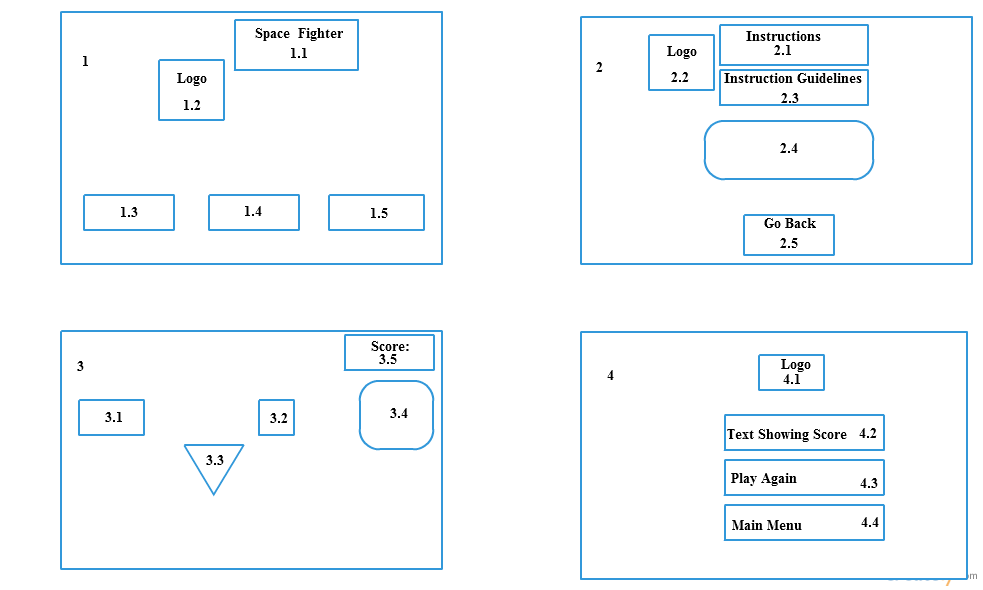
4.1 Game Over which is a GUI Text.

4.2 Text showing score which are GUI Text.

4.3 Retry Button which is a GUI Button.

4.4 Main Menu Button which is a GUI Button.

* **Space Fighter**

****

1. The main menu screen is an image of type PNG.

1.1 Space Fighter which is a GUI Text.

1.2 Logo which is an image of type PNG.

1.3 Play Button which is a GUI Button.

1.4 Instructions Button which is GUI Button.

1.5 Quit Button which is a GUI Button.

2. The instructions screen is an image of type PNG.

2.1 Instructions which is a GUI Text.

2.2 Logo which is an image of type PNG.

* 1. Instructions guidelines which is GUI Text.
  2. Keyboard image which is an image of type GIF.
  3. Go Back Button which is a GUI Button.

3. The game screen is an image of type PNG.

3.1 Rocket which is an image of type PNG and has Sprite Renderer, Box Collider

2D, Rigid Body 2D, appropriate scripts attached.

3.2 Poulpi (enemy) which is an image of type PNG and has Sprite Renderer, Box

Collider 2D, Rigid Body 2D and appropriate scripts attached.

3.3 Platforms which is an image of type PNG.

3.4 Boss (enemy) is an image of type PNG and has Sprite Renderer, Box Collider

2D, Rigid Body 2D, appropriate scripts and Animator attached.

3.5 Score which is a GUI Text.

4. The game over screen is an image of type PNG.

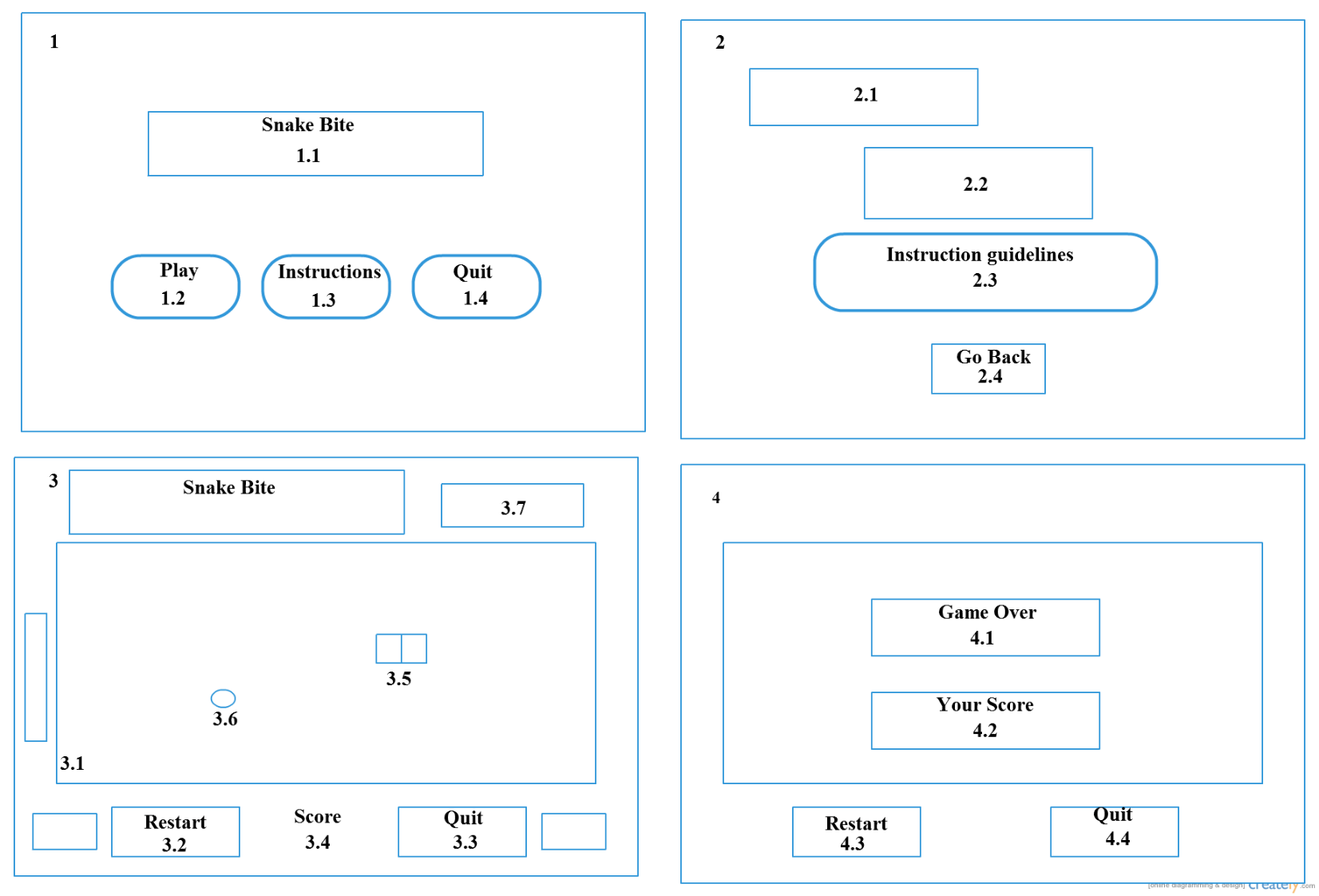
4.1 Logo which is an image of type PNG.

4.2 Text showing score which is a GUI Text.

4.3 Play Again Button which is a GUI Button.

4.4 Main Menu Button which is a GUI Button.

* **Snake Bite**

****

1. The main menu screen is an image of type JPEG.

1.1 Snake Bite which is a GUI Text.

1.2 Play Button which is a GUI Button.

1.3 Instructions Button which is a GUI Button.

1.4 Quit Button which is a GUI Button.

2. The instructions screen is an image of type JPEG.

2.1 Instructions which is an image of type PNG.

2.2 Keyboard image which is an image of type GIF.

2.3 Instructions guidelines which is a GUI Text.

2.4 Go Back Button which is a GUI Button.

3. The game screen is an image of type PNG.

3.1 Wall which is the boundary and has Sprite Renderer and Box Collider 2D.

3.2 Restart Button which is a GUI Button.

3.3 Quit Button which is a GUI Button.

3.4 Score which is a GUI Text.

3.5 Snake is an Image of type PNG which has appropriate scripts attached.

3.6 Food is an Image of type PNG having prefab properties.

* 1. Snake image of type PNG.

4. The game over screen is an image of type PNG.

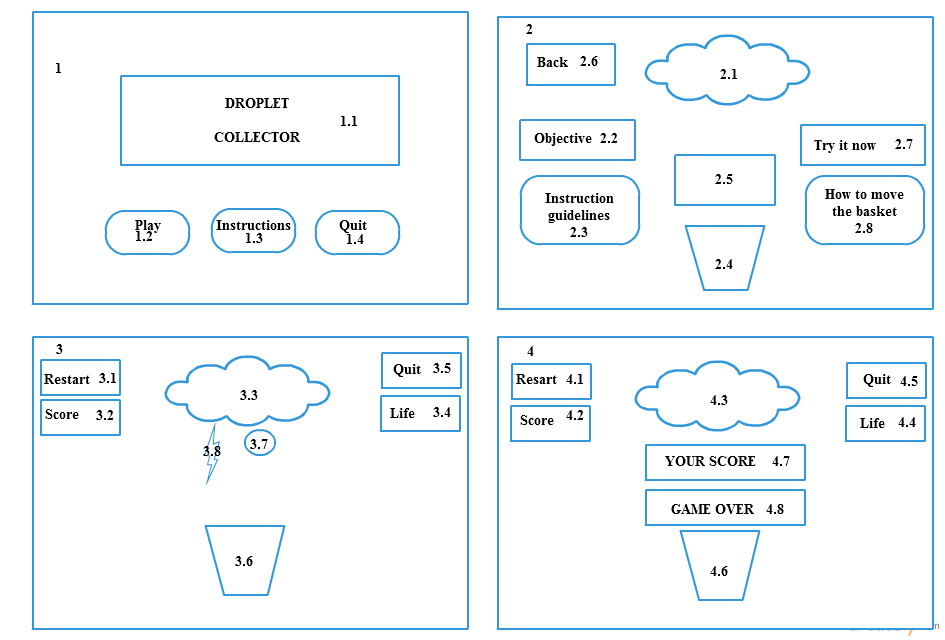
4.1 Game over which is a GUI Text.

4.2 Text showing score which is a GUI Text.

4.3 Restart Button which is a GUI Button.

4.4 Quit Button which is a GUI Button.

* **Droplet Collector**

****

1. The main menu screen is an image of type JPEG.

1.1 Moving droplets heading which are images of type JPEG with script attached.

1.2 Play Button which is a GUI Button.

1.3 Instructions Button which is GUI Button.

1.4 Quit Button which is a GUI Button.

2. The instructions screen is an image of type JPEG.

2.1 Cloud which is an image of type PNG.

2.2 Objective which is GUI Text.

2.3 Instructions guidelines which is GUI Text.

2.4 Bucket which has Sprite Renderer, Box Collider 2D, Rigidbody 2D,

appropriate script attached.

2.5 Keyboard image which is an image of type GIF.

2.6 Go Back Button which is a GUI Button.

2.7 Try it now which is a GUI Text.

2.8 How to move the basket which is a GUI Text.

3. The game screen is an image of type JPEG.

3.1 Restart Button which is a GUI Button.

3.2 Score which is a GUI Text.

3.3 Cloud which is an image of type PNG.

3.4 Life which is a GUI Text.

3.5 Quit which is a GUI Button.

3.6 Bucket which has Sprite Renderer, Box Collider 2D, Rigidbody 2D,

appropriate script attached.

3.7 Droplet which is an image of type PNG and has script attached.

3.8 Thunder which is an image of type PNG and has script attached.

4. The game over screen is an image of type JPEG.

4.1 Restart Button which is a GUI Button.

4.2 Score which is a GUI Text.

4.3 Cloud which is an image of type PNG.

4.4 Life which is GUI Text.

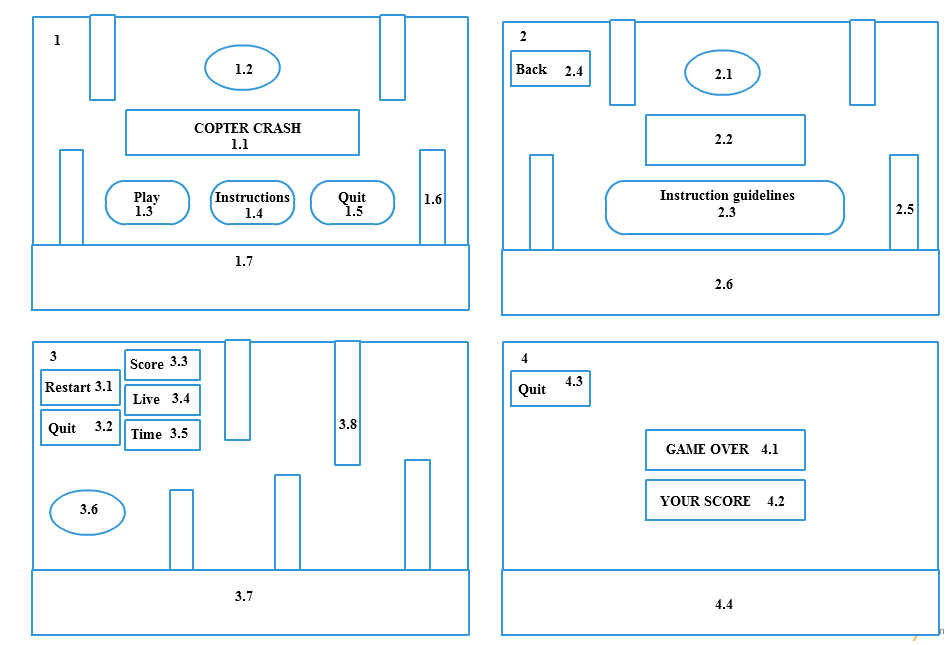
4.5 Quit which is a GUI Button.

4.6 Bucket which has Sprite Renderer, Box Collider 2D, Rigidbody 2D.

4.7 Your Score which is a GUI Text.

4.8 Game Over which is a GUI Text.

* **Copter Crash**

****

1. The main menu screen is a Skybox image.
   1. Copter Crash which is a GUI Text.
   2. Plane which has Sprite Renderer, Circle Collider 2D, Rigidbody 2D and appropriate script attached.

1.3 Play Button which is GUI Button.

1.4 Instructions Button which is GUI Button.

1.5 Quit Button which is GUI Button.

1.6 Obstacles which have Sprite Renderer, Rigidbody 2D and appropriate script

attached.

* 1. Ground which has Sprite Renderer and Box Collider 2D attached.

2. The instructions screen is a Skybox image.

2.1 Plane which is an image of type PNG.

2.2 Keyboard image which is an image of type GIF.

2.3 Instructions guidelines which is GUI Text.

2.4 Go Back Button which is a GUI Button.

2.5 Obstacles which have Sprite Renderer, Rigidbody 2D and appropriate script

attached.

2.6 Ground which has Sprite Renderer and Box Collider 2D attached.

3. The game screen is a Skybox image.

3.1 Restart Button which is a GUI Button.

3.2 Quit which is a GUI Button.

3.3 Score which is a GUI Text.

3.4 Life which is a GUI Text.

3.5 Time which is a GUI Text.

3.6 Plane which has Sprite Renderer, Circle Collider 2D, Rigidbody 2D, audio

source and appropriate script attached.

3.7 Ground which has Sprite Renderer and Box Collider 2D attached.

3.8 Obstacles which have Sprite Renderer, Rigidbody 2D and appropriate script

attached.

4. The game over screen is a Skybox image.

4.1 Game Over which is a GUI Text.

4.2 Score which is a GUI Text.

4.3 Quit which is a GUI Button.

4.4 Ground which is image of type JPEG.

**Conclusion**

As with any game development tool, the key to success with Unity is iterative development. One has to build in manageable increments – be ambitious, by all means, but be ambitious in small chunks, and arrange those chunks such that, even if one falls short of their ultimate ambition, one will at least wind up with a coherent product. Get the most crucial elements in first: have an idea in mind of one’s ‘minimum viable product’, the simplest, most bare-bones thing one could possibly create and still feel as though one has achieved something worthwhile. Get to that minimum viable project before moving on to larger ambitions.

An attempt to bring platform games into 3D. It brings an enjoyable experience to the game lovers. The implementation of the techniques is illustrated through the PC Game and described in detail with the complete code and Gaming Interface. Therefore, this project results in a successful definition of the instructions for the development techniques and a functional PC Based Game Application.