**Unity 3D Learning Portfolio**

**By Ajay Balakrishnan**

**About the game – DodgeRoller**

**GitHub Link to the Game -**

The rules:

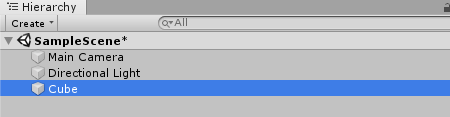
You must avoid the obstacles by moving the ball side to side and reach the end point of the level. Very simple rules.

If you hit the obstacle or fall off the edge, you will restart from the current level.

There are only 2 levels 🡪 The second level is more challenging as the obstacles are placed more awkwardly.

**Lesson 1**

* Creating shapes and objects in the default space.



Click on the create tab and a list of options will be given as to what object you want to create. I used a 3D cube to create my ground where the ball rolls on. Then I create a sphere following the same procedure.

* Learning how to navigate around Unity using shortcuts.

“q” – Is the hand tool which allows you to pan around the scene.



“w”- Is the move tool which allows you to move objects in the scene.



“e” – Is the rotate tool which allows you to rotate objects in the scene.

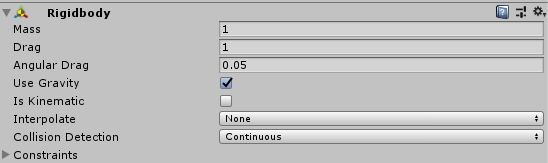


“r” – Is the scale tool which allows you to rescale an object in the scene.

**Lesson 2**

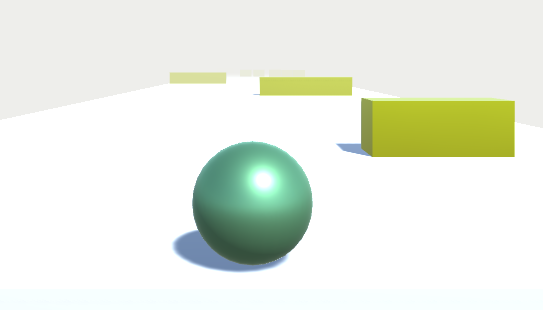
* Applying physics to objects and rigid bodies.

In the second session, I learnt how to add rigid bodies to the sphere that I created so that I can add gravity to the object and program movement. You add a new component called rigid bodies to the object.This allows you to modify the mass, drag and the overall physics of the object.



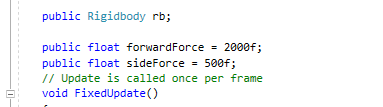
* Creating the scene for the game

I wanted to go for a basic look but effective look in my game. I went for an odd choice of colours to attract the eye.



**Lesson 3**

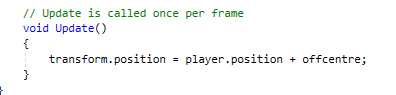
* How to code movement for my sphere object



Forces were used to code the movement of my object and in the code you could see the different variables for all the movement.

**Lesson 4**

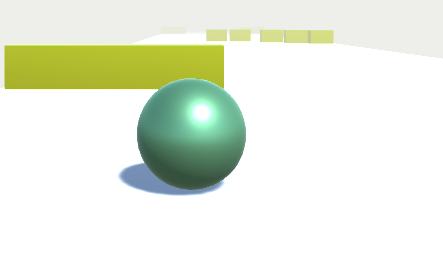
* Figuring out how to make the camera follow along with the object.



This was easily done by coding the camera movement along with the position of the object. However, this took some time as doing that gave me a first person view of the object. Changing the z axis and adding a vector to the original position allow me to track the object in third person this time.

**Lesson 5**

* Collision detection and allowing the sphere to come to a halt when it collides.

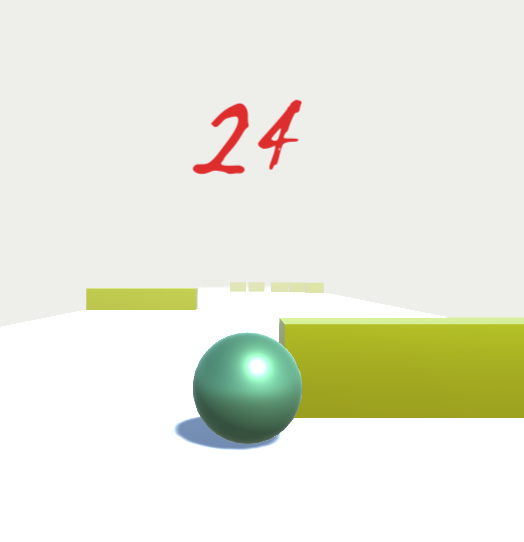


As the ball collides with the obstacle the movement of the object becomes false, making the ball slow down and eventually come to a stop.

Also I focused on making the movement of the ball more smoother. I figured that coding it in update meant that the ball would move at different speeds depending on the frame rate of the PC. Therefore to balance out the speed, I introduced the (Time.deltaTime), which did work but I had to adjust the forward speed as the speed had slown down.

**Lesson 6**

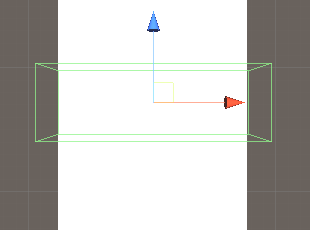
* Introducing scores and keeping track of score. I implemented the score, purely for the fact that the player just see how far in the level they had got because it wouldn’t really matter if the complete the level. They may have a rough idea of how far they were from the end.



The score was put using the position of the ball as the ball is travelling forward from a position of zero, so I thought tracking the position of the ball would be a great way to implement scores.

**Lesson 7**

* I used triggers to set the endpoint of each level, therefore when the person reaches the end, the game cuts to the next level or the end credits, based on which scene the person is currently in.



I used a cube as a trigger, by turning off the mesh renderer so that the ball cannot collide with it but can trigger it by passing through it.

* I was designing the finishing of the game. I had put in 2 levels into the game. As the game was very simple looking, I just wanted to have a simple way to end it as well.



So I just did a credits page with a quit button.

**Lesson 8**

* The game was being touched up and finished.
* A main menu was added.

