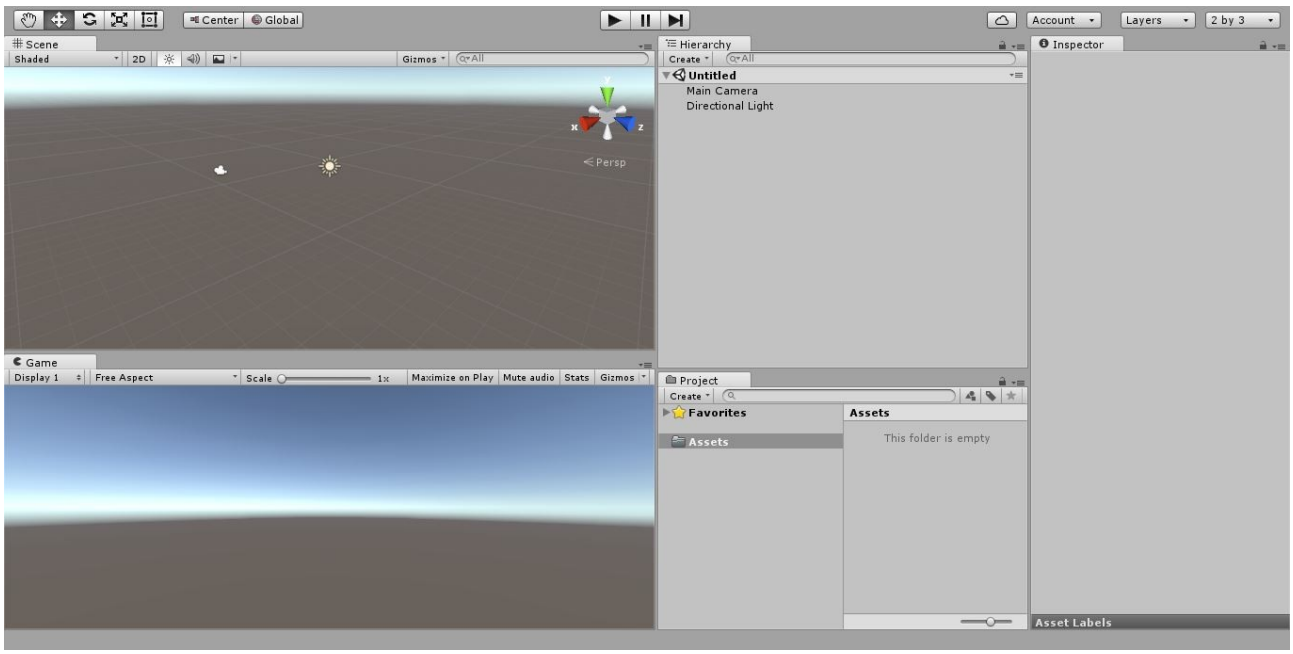
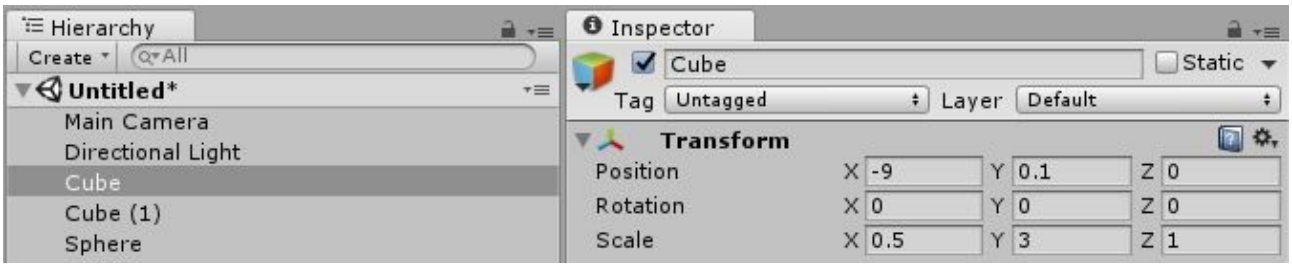
Pong Tutorial

Starting with Unity3d, the engine should initially appear like that:



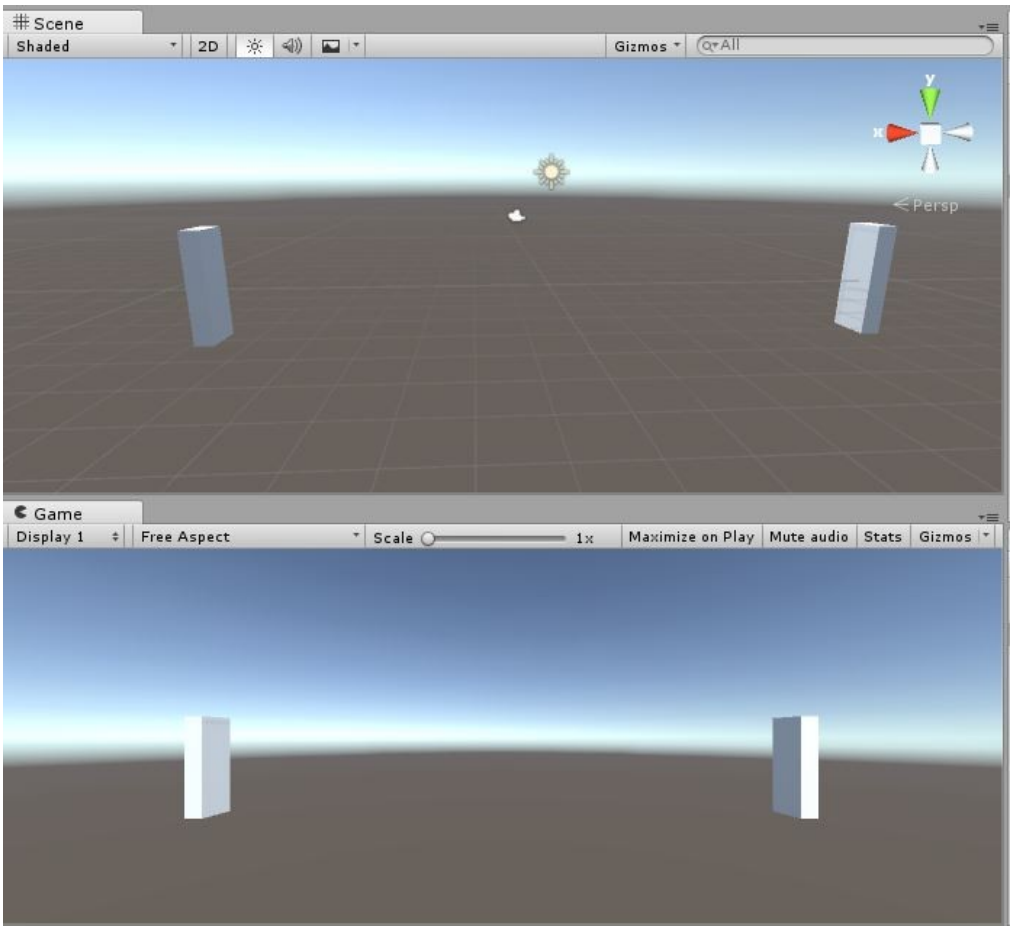
This is a sample layout, 2 by 3. You can adjust it to your liking by dragging the sections to the desired place or choose one of the default layout settings (located on the top right corner).

Our goal is to make a simple pong game. What kind of shapes does one need to do so? Two paddles, two walls for the ball to bounce on and of course the ball, therefore we need a total of 5 shapes. We can classify better our needs, since we need 2xPaddless, 2xWalls and 1 ball. So at the beginning we will make 1 cube which will be duplicated to create 2 paddles, 1 more cube which will be duplicated to create 2 walls and 1 sphere to create the ball. Locate “Hierarchy” and click “Create” > “3D Object” > “Cube” twice, and “Create” > “3D Object” > “Sphere” once. Clicking on a cube, you can see the object’s attributes appearing under the “Inspector”. There are four sections; Transform, Cube (Mesh Filter), Box Collider and Mesh Renderer, as well as an extra section named “Default-Material”. Focusing on the Transform section, you can see the cube’s **position**, **rotation** and **scale** options. You can freely edit these and change the shape and location of the cube.



At the position enter X: -9, Y: 1, Z: 0, make sure that at the rotation every coordinate is set to 0 and at the scale enter X: 0.5, Y: 3, Z: 1, as shown above. While changing these numbers, you can see at the Scene to the left the cube being altered in real time. At the cube's inspector you can see an option for its name, proceed to rename it to "Paddle1". At the Game section, under the Scene, you can see that Paddle1 is on the left side, so we need the next paddle to be at the exact opposite position. Right click on Paddle1 and select "Duplicate". Change from the new 3D object's inspector the name from Paddle1 (1) to Paddle2. The only thing needed to be changed at this new paddle, is the Transform>Position>X coordinate. The Paddle1's was -9, so we will put this at X: 9.

So far here are our two paddles:



Keep in mind that the Scene section is where you can freely move objects by hand in case you do not want to use certain coordinates, and the Game section is an overview of what the game will look like from the player's aspect once it starts.

Now time to alter the sphere. Change its transform to X: 0, Y: 1, X: 0 and leave the rest at the default settings. This object however is a little more complicated than the paddles. Having selected the Sphere, click on the tabs menu and select Component > Physics > Rigidbody.

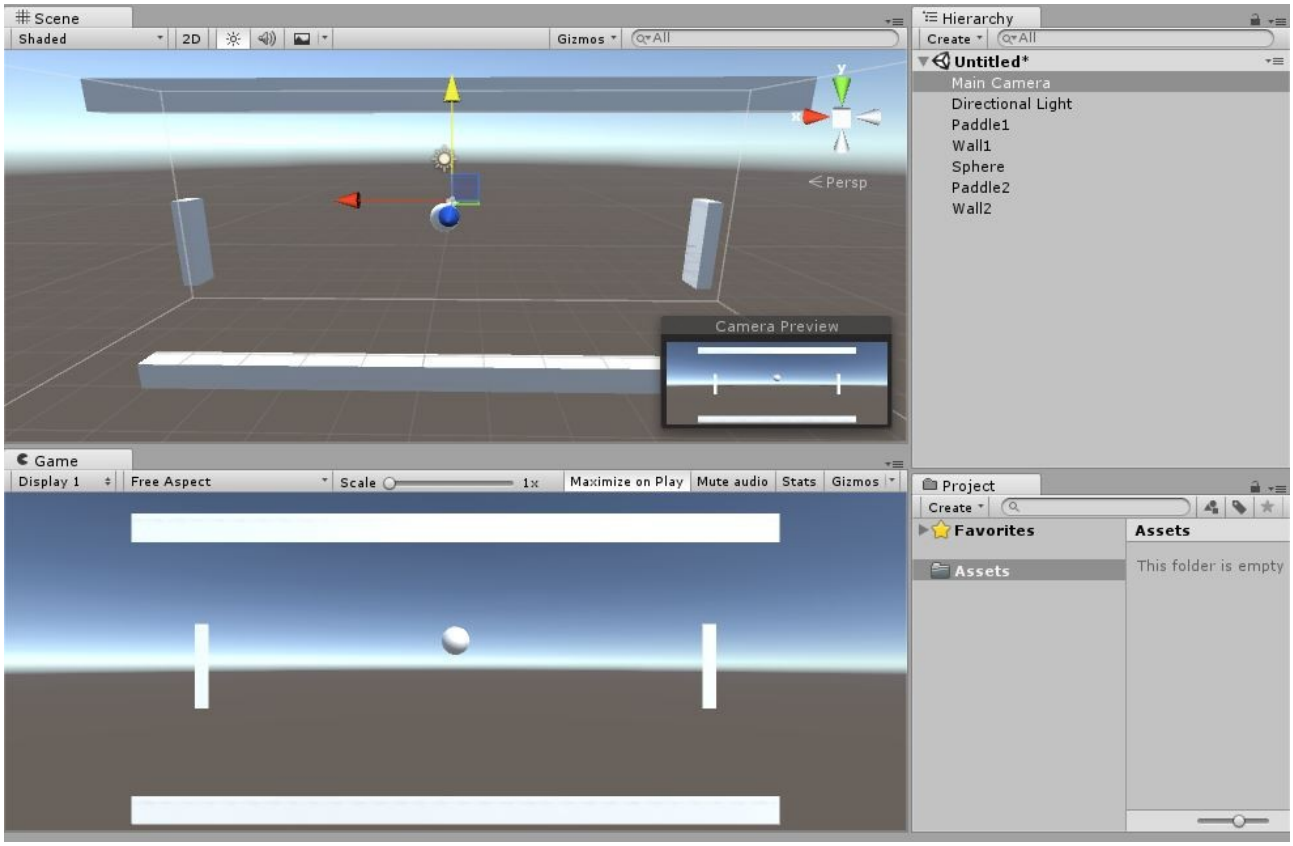
What is a Rigidbody? Quoting Unity3D Documentation:
“Adding a Rigidbody component to an object will put its motion under the control of Unity's physics engine. Even without adding any code, **a Rigidbody object will be pulled downward by gravity** and will react to collisions with incoming objects if the right Collider component is also present.”

The bold part is something we definitely do not want, since it is gonna drag our ball down ruining the game. At the sphere’s inspector, locate the new Rigidbody section and untick the “Use Gravity” option which is enabled by default. The ball is now ready.

Moving on to the last object we will alternate, select Cube (1) and change its Transform attributes to Position: X: 0, Y: -5, Z:0 and Scale: X: 23, Y: 1, Z: 1, leaving rotation as it is. Rename it to Wall1, right click on it through the Hierarchy and duplicate it. Rename the new 3D object to Wall2, and change just the Position>Y option to 5 from -5.

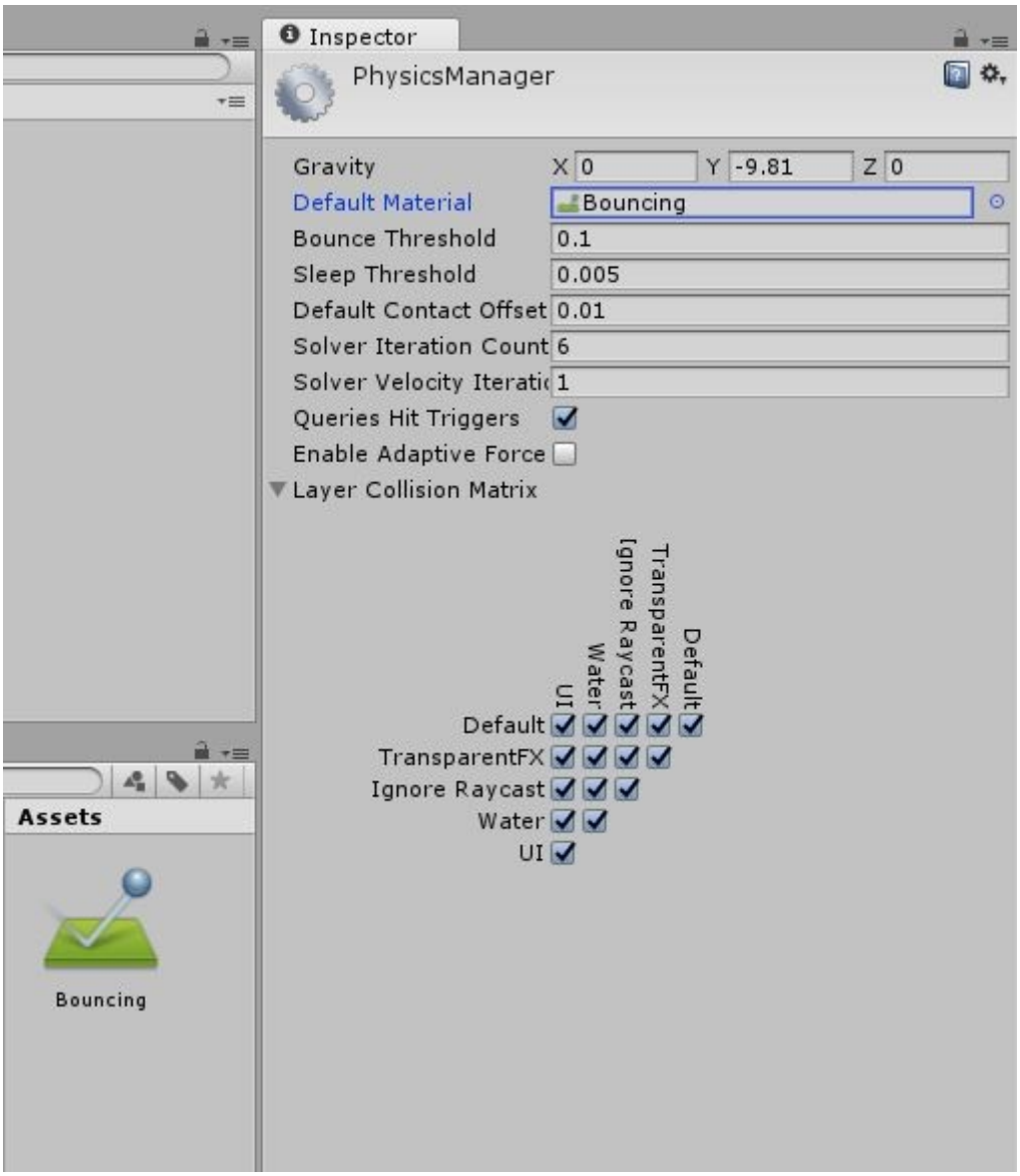
You should have noticed that since the beginning there were two more objects in the Hierarchy that we did not alternate in any way. The one is the Directional Light which is making everything visible, and the other one is the Main Camera. The camera is basically the player’s optical perspective. Select the “Main Camera” and locate the “Camera” section. Find the “Projection” option and change it to orthographic and then change the size to 6.

This is how the scene should look after creating and altering all these objects.



We are done with creating and changing shapes, we will move on with physical attributes and programming, otherwise we just have 5 shapes that will never react to anything.

At the Program section below Hierarchy, select Assets and right click > Create > Physic Material. Rename it to “Bouncing” because this is what we will use that for, defining parameters considering the bouncing of the ball. Click on the “Bouncing” and from its inspector change Dynamic Friction to 0, Static Friction to 0, Bounciness to 1, Friction Combine to Minimum and Bounce Combine to Maximum. Next locate on the toolbar Edit > Project Settings > Physics. Drag and drop Bouncing on the “Default Material” option and also change the Bounce Threshold to 0.1.



It is now time for scripts! Right click on Assets > Create > C# Script twice. Name the first script “Ball” and the second “PaddleController”. While at the Scene section, drag and drop the Ball script on the sphere and the PaddleController on both paddles. Double click on the scripts and wait until the Unity IDE MonoDevelop appears with both of the scripts open.

For the Ball.cs :

```
using UnityEngine;
using System.Collections;

public class Ball : MonoBehaviour {

    float speedX;
    float speedY;

    // Use this for initialization
    void Start () {
        speedX = Random.Range (0, 2) == 0 ? -1 : 1;    //assigning speed
values on both axis for the ball
        speedY = Random.Range (0, 2) == 0 ? -1 : 1;

        GetComponent<Rigidbody> ().velocity = new Vector3 (Random.Range (
5, 10) * speedX, Random.Range (5, 10) * speedY, 0);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

The only thing we care about is assigning speed on both axis for the ball and then let it bounce accordingly. We want to keep track of the ball's speed and we only interfere with the initial parameters, therefore there is no need to write something in the Update function.

For the PaddleController.cs:

```
using UnityEngine;
using System.Collections;

public class PaddleControler : MonoBehaviour {

    // Use this for initialization
    void Start () {
    }

    // Update is called once per frame
    void Update () {
      float speed = .2f;            //paddle's movement speed
      if (gameObject.name == "Paddle1")
        transform.Translate (0, Input.GetAxis ("Horizontal") * speed, 0);
      else
        transform.Translate (0, Input.GetAxis ("Vertical") * speed, 0);
    }
}
```

So what is exactly happening in this if statement? We want this game to be played by two players on the same keyboard. Instead of assigning keys depending on the movement the player chooses in each update loop, we can assign **axis movements**.

From Unity3d Documentation:

“public void Translate(float x, float y, float z, Space relativeTo = Space.Self);

Description

Moves the transform by `x` along the x axis, `y` along the y axis, and `z` along the z axis.

If `relativeTo` is left out or set to Space.Self the movement is applied relative to the transform's local axes.”

Hence, the Translate function in the if statement above is going to adjust the Horizontal axis as movement on the y axis. So we can have two vertical movements for the paddles assigned with just 2 lines of code.

By default, the W S keys are for vertical movement therefore they are ok for the one paddle, while the keys for horizontal movement are A D. We need one more twitch on the game keys in order to make it easier for the Paddle1 player. From the Toolbar Edit > Project Settings > Input and expand Axes > Horizontal. At the option alt positive button change “d” into “q”.

Save both of your scripts and hit play. Control one paddle with q-a and the other one with w-s.

You just created your first pong game!