

Unity Intro!

To follow along and do this yourself you will need Unity Installed and working. (I'll be using 5.2.4)

Some assets we will use: (nothing fancy)

<http://csciclub.github.io/jeff/presentation%20assets.zip>

Everything we will do can be found at:

<http://csciclub.github.io/jeff/ball.zip>

<http://csciclub.github.io/jeff/badmonty.zip>

(5.2.4 projects made on windows 10)

What is Unity?!?!?!?!?

- ▶ Unity is a very awesome and free game engine that can be used to build games, apps, models, simulations ect.

What is Unity?!?!?!?!?

- ▶ Unity is a very awesome and free game engine that can be used to build games, apps, models, simulations ect.

Cool things about Unity

- ▶ Super fast deployment to many platforms
 - ▶ Just select which platform you want to build for and Unity takes care of it after you set it up.
 - ▶ Android, IOS, windows, mac, linux, others like xbox
- ▶ Huge API with documentation that is extremely easy to read and find
- ▶ Out of the box API integration with MonoDevelop.
- ▶ Script in C#/javascript
- ▶ Support for many common asset file types like .blend
- ▶ Oculus rift support



Lets Make something!

- ▶ **Start Unity**
- ▶ **New Project**
- ▶ **Name it something**
- ▶ **Click 3D**
- ▶ **Create Project**

Lets Make something!

- ▶ Start Unity
- ▶ New Project
- ▶ Name it something
- ▶ Click 3D
- ▶ Create Project

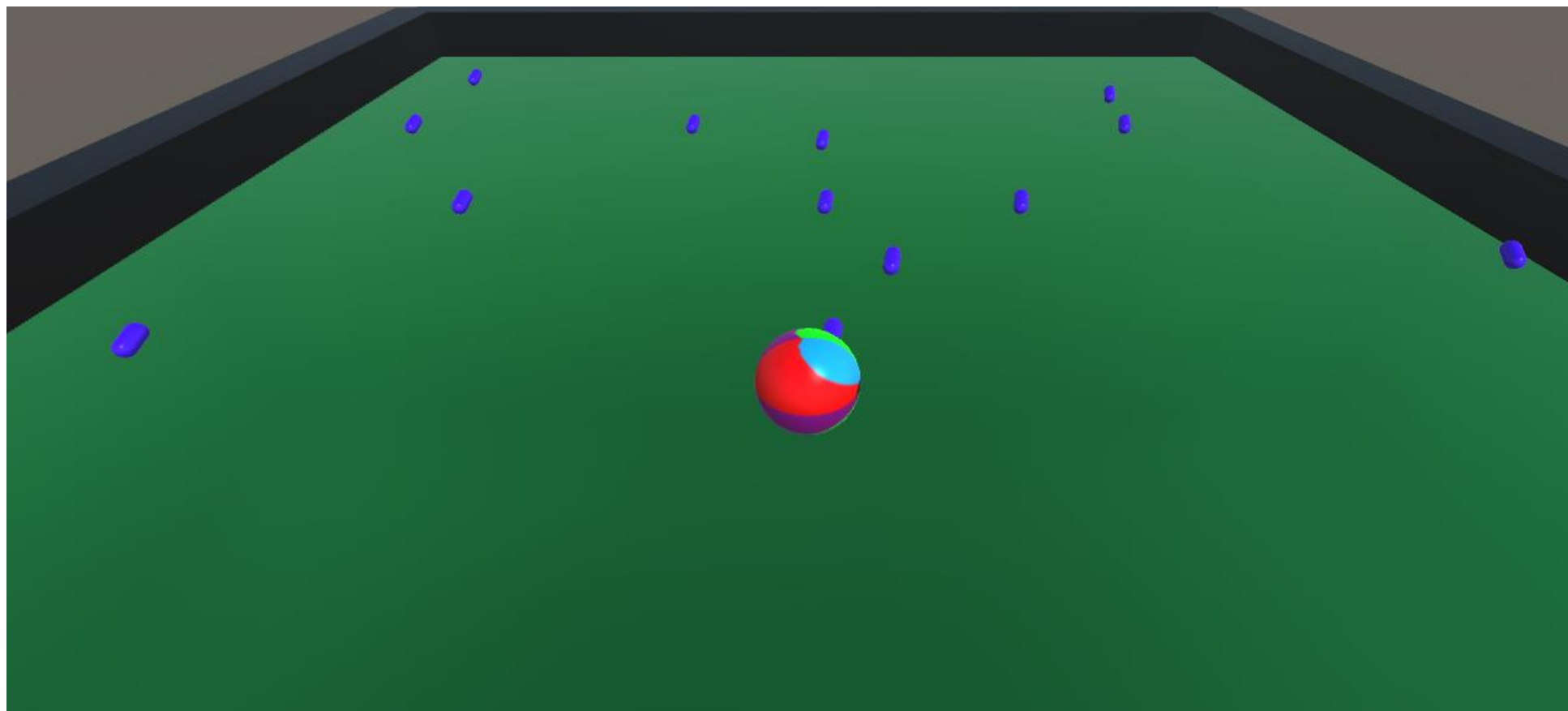


Interface

▶ Many sub-windows and tabs

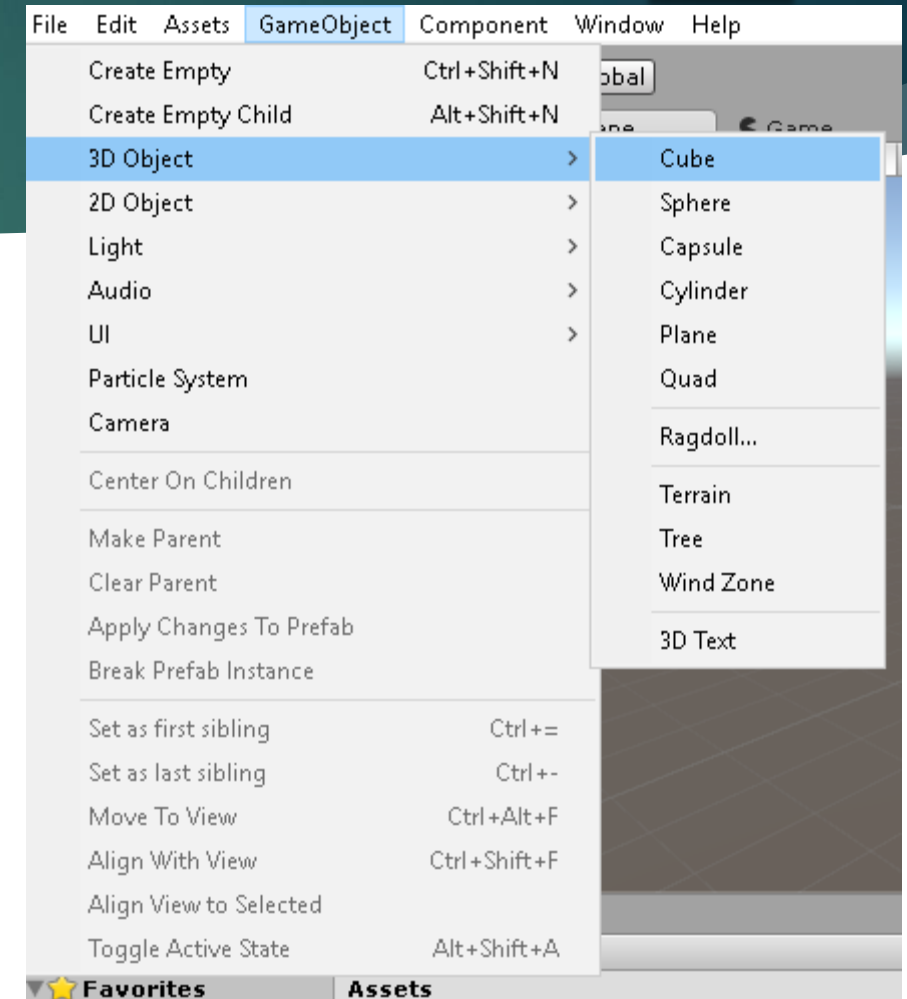
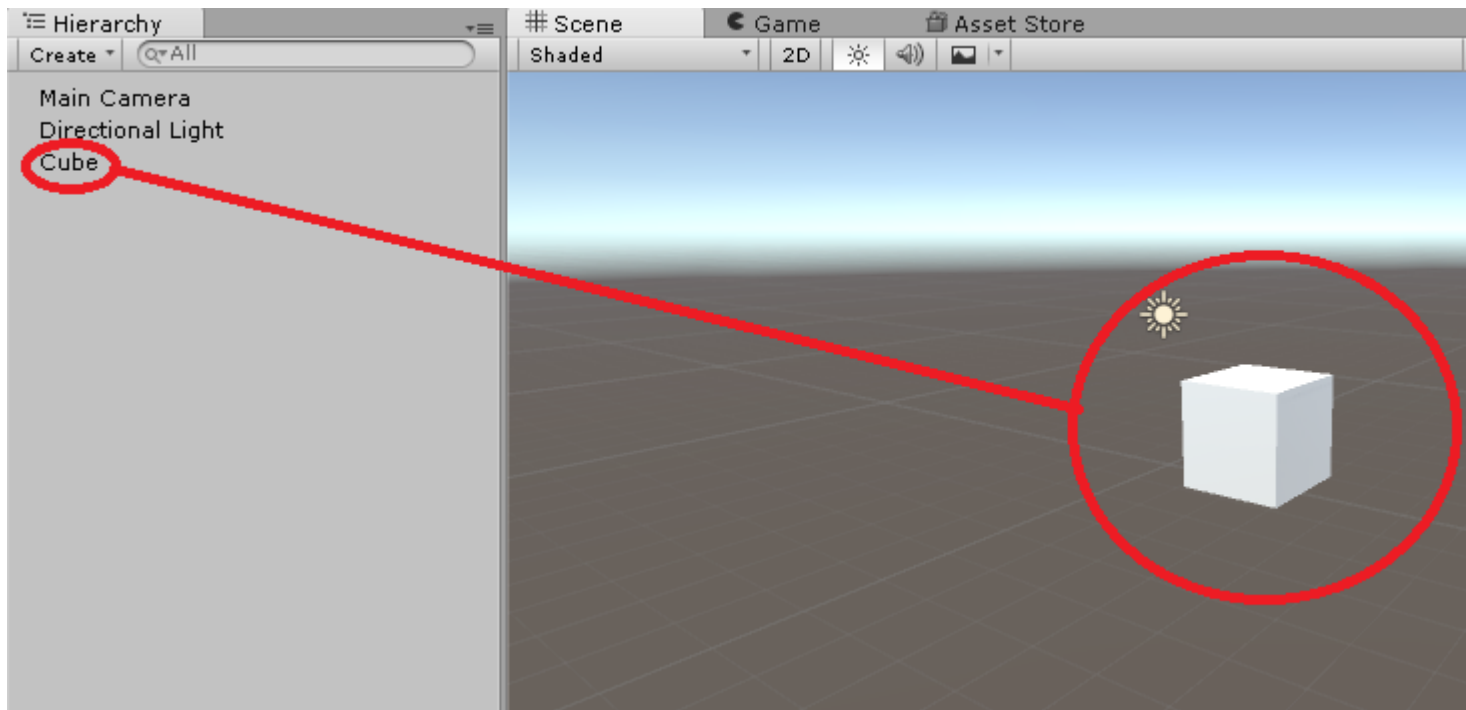
- ▶ Scene – This is where you place things in the 3d space
- ▶ Game – The view from your main camera
- ▶ Hierarchy – The list of every item in your scene
- ▶ Project – File explorer for all files associated in your program. (images, scripts, animations, 3d models, sounds, videos, Pre made game objects)
- ▶ Inspector – Displays information about the currently selected item in the hierarchy
- ▶ Console – displays messages, like errors, warnings, printed statements from scripts.
- ▶ There are some others for specific things like building animations, flow charts, and modifying sprites.

Our First Game!



1 create the surface

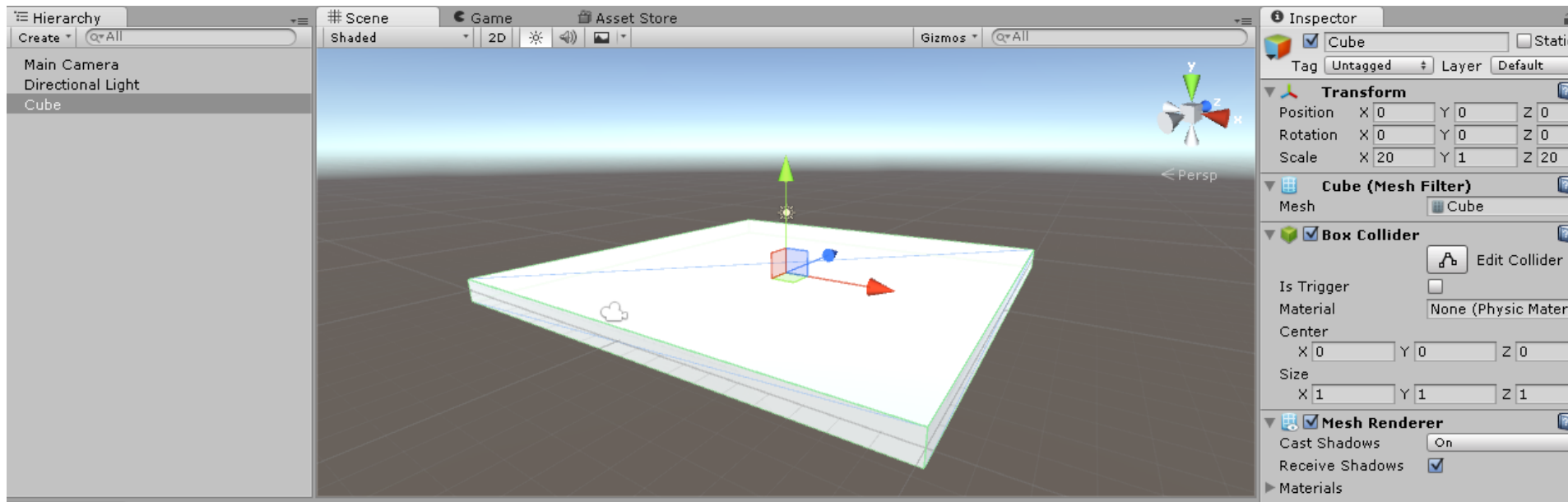
- ▶ Select GameObject → 3D Object → Cube
- ▶ A 1x1x1 (unit unicube) should now be in your scene!



Create a 20x20x1 platform

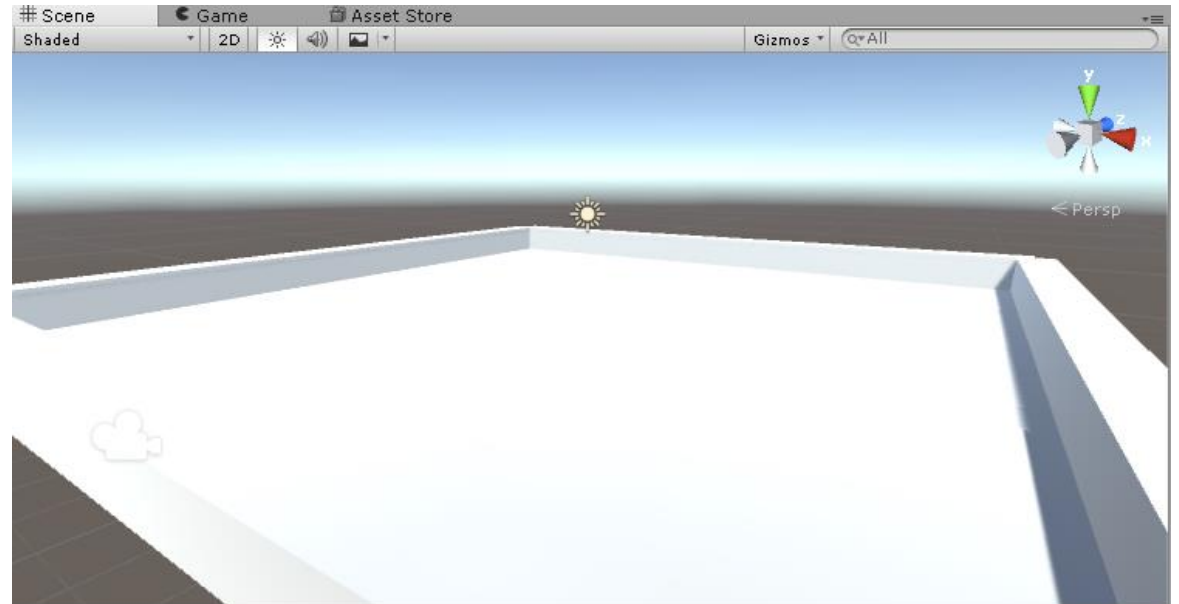
- ▶ In your Scene, or Hierarchy tabs select the newly created cube and look at the inspector. Under “Transform” scale the cube to (x=20, y=1, z=20)
- ▶ Also Under “Position” move the cube to (x=0, y=0, z=0).

(if you want to focus on an object in the scene, select the object and with your mouse hovering over the scene window, and press F)



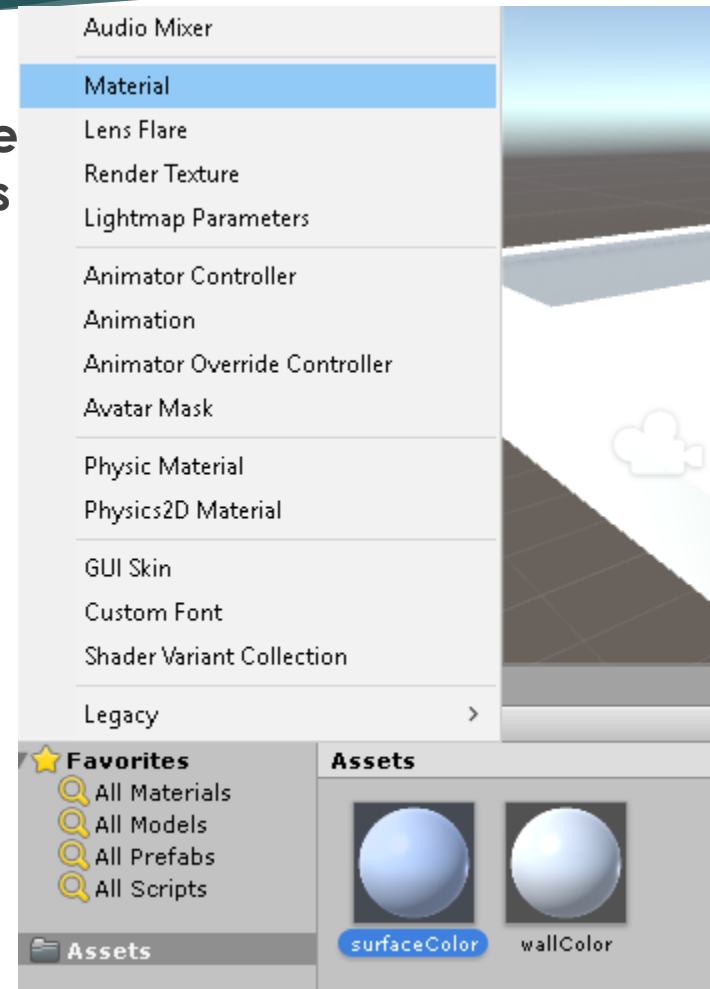
Walls

- ▶ Similarly we will create 4, 20x1x1 walls around our platform.
- ▶ Create 4 more cubes, scale each to (20x1x1), and alter their transforms so that they are around the platform.
- ▶ Objects can be copy/pasted/cloned
- ▶ Cube 1- $P(0,1,10)$ $R(0,0,0)$ $S(20,1,1)$
- ▶ Cube2 – $P(0,1,-10)$ $R(0,0,0)$ $S(20,1,1)$
- ▶ Cube 3- $P(10,1,0)$ $R(0,90,0)$ $S(20,1,1)$
- ▶ Cube 4- $P(-10,1,0)$ $R(0,90,0)$ $S(20,1,1)$



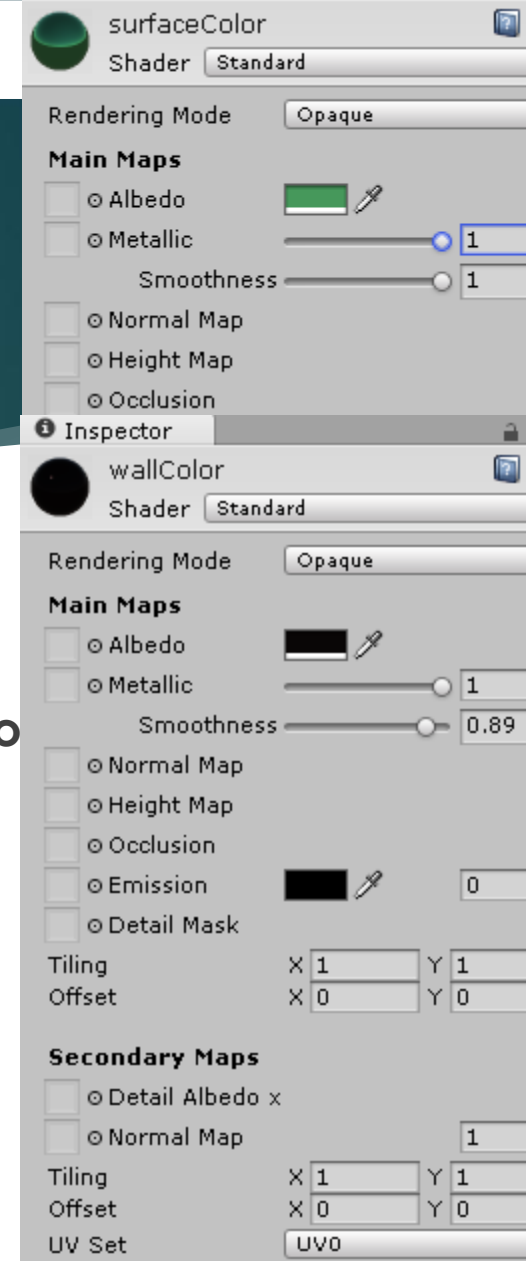
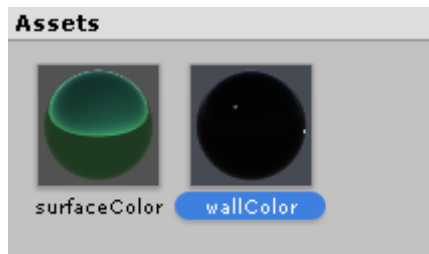
Materials

- ▶ We should probably make use of the sweet graphics engine. Here add colors (materials) to each of our walls and surface. Materials are what we use to wrap 3D models. Here we will just use basic colors.
- ▶ In the top left of the “Project” tab, select create→Material
- ▶ Name the material something like “wall material”
- ▶ And create another material for the surface.



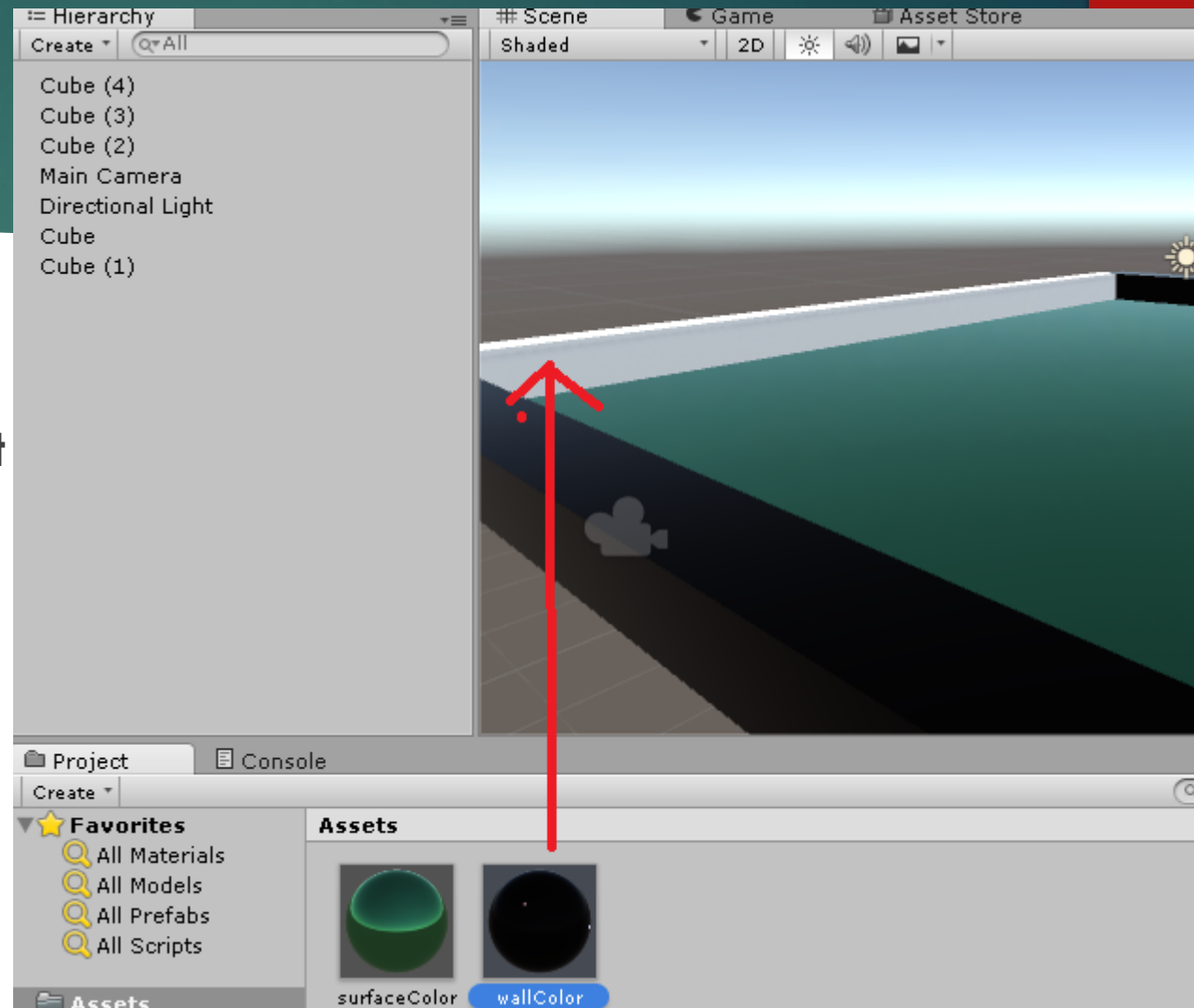
Materials

- ▶ Select the wall material and take a look at the inspector window
- ▶ Here we can use some tools to create edit the material we just created, lots of options for things like colors , shininess, and images that you want to put over your models.
- ▶ Choose two colors you like =)



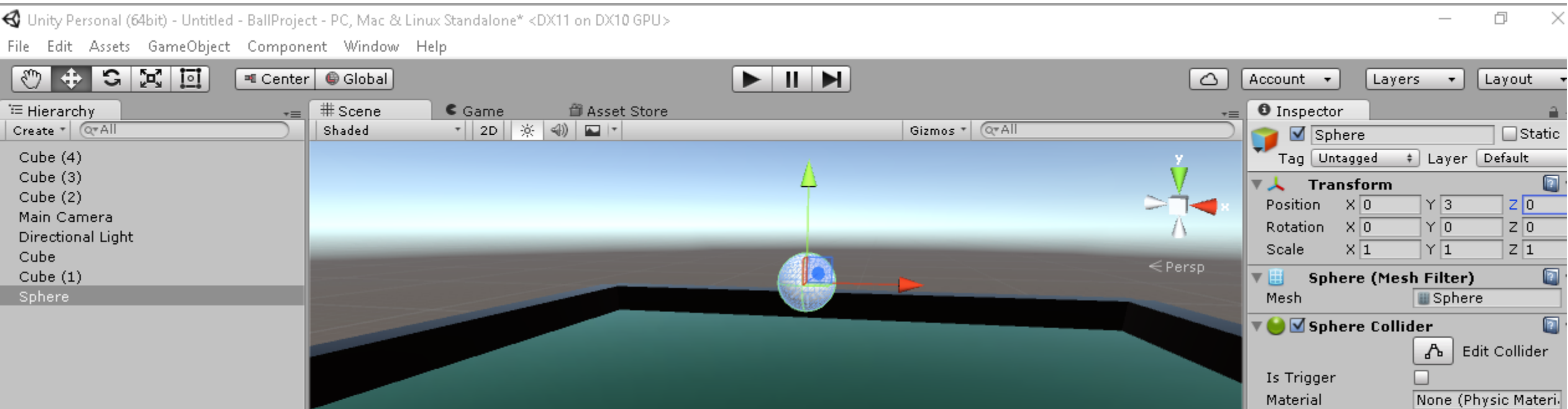
Materials

- ▶ Drag these materials to the scene tab and drop them over the objects that you want to apply them to!



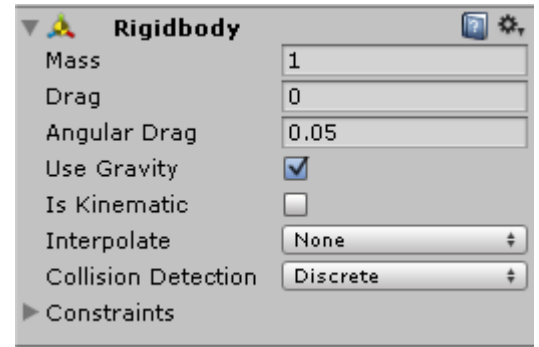
Creating the player!

- ▶ We're going to make a ball that will roll around when we control it
- ▶ Select GameObject → 3D Objects → Sphere
- ▶ Select the sphere and reposition it to (0,3,0)



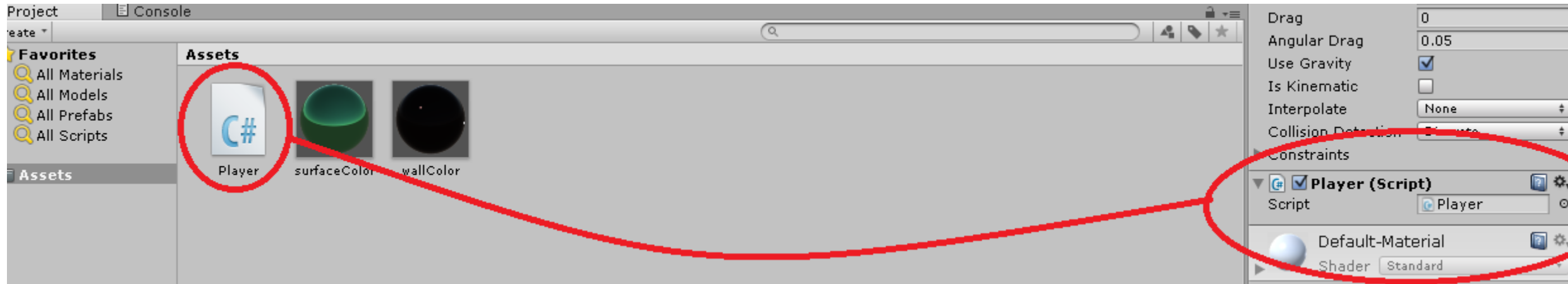
Adding the Player

- ▶ If you press the Play button up top, you will see the view from the main camera, nothing much is happening here.
- ▶ Un-play and select your sphere. We want the ball to fall when the game is live. So we need to make the ball an object that behaves according to Unity's physics engine (you could roll your own physics too though)
- ▶ Notice in the inspector you have several different components: Transform, Sphere mesh filter, sphere collider, mesh render and material. These are pretty self explanatory.
- ▶ Select Add component → Physics → Rigidbody
- ▶ Your ball will now have a Rigid Body component, which makes it obey Unity's laws of physics! **Press Play and see! =)**
- ▶ Your ball stops because the box colliders of the floor and the sphere collider of the ball are colliding.

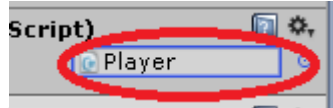


Adding the player

- ▶ So now we have this ball that falls. The coolest thing that has happened in 14 slides..
- ▶ Now lets figure out some controls!
- ▶ Select the ball, AddComponent → New Script → name, csharp, create and add.



Unity Scripting

- ▶ Finally some code stuff
- ▶ Double click on the script you created : 
- ▶ This should open MonoDevelop if you let it install the way it wants to, idk about linux/Apply but if not any text editor will suffice, they're just less integrated



```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
```

Scripts

- ▶ The script will automatically create a new class that is the name of your script, include the default libraries, with a Start() and a Update() function these are inherited from MonoBehaviour.
- ▶ In the start function, you put everything you want to happen when the Game Object is first created or the play button is pressed.
- ▶ The Update function is called once per frame, there is also a “FixedUpdate” function you could use instead that is synchronized with the draw rate I think.
- ▶ In our start we will initialize things like speed and getting components that we will be manipulating.
- ▶ In our update function we will do all of our things that will be happening over and over again, like checking for movement keys being pressed!

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11     // Update is called once per frame
12     void Update () {
13
14     }
15 }
16
```

Scripts

Attaching a script to a game object makes it live, and it will be running as long as that game object exists in the scene. The script can immediately access the game object it is attached to by **gameObject** which is of type `GameObject`.

For example, `gameObject.transform.position;` is the 3d position of the object that has the script attached to it which is of type `Vector3`.

Members of vector 3 can be accessed by `.x .y .z`

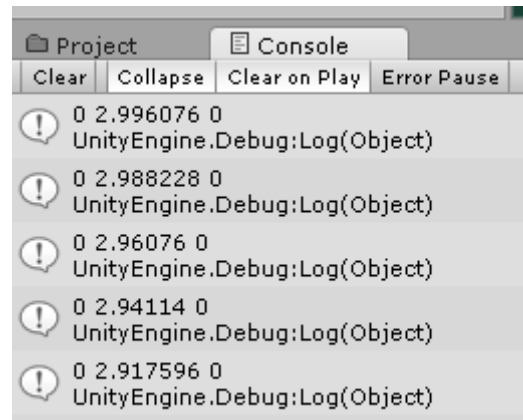
For example `Vector3 pos = gameObject.transform.position;`

`Debug.log(pos.x+ " " +pos.y+ " " + pos.z + "\n");`

Scripts

- ▶ Type and save the code to the right and press the play button in Unity.
- ▶ In the console window you should see the position of the sphere being spammed.
- ▶ One per frame is pretty frequently so you need to be careful about what goes into the Update function if you want decent performance.
- ▶ Next let's make the ball move in the x direction.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour {
5     Vector3 pos;
6     // Use this for initialization
7     void Start () {
8
9     }
10    // Update is called once per frame
11    void Update () {
12        pos = gameObject.transform.position;
13        Debug.Log (pos.x + " " + pos.y + " " + pos.z);
14    }
15 }
16
```



Scripts

- ▶ You can directly assign the `gameObject.transform.position`, however that might look choppy so we will apply a Force to the ball.
- ▶ Forces can be applied to the Rigidbody component of the `gameObject`.
- ▶ To get a specific component of a `gameObject` we will use the `GetComponent<Template T>()` function.
- ▶ We will get the Rigidbody component by:
- ▶ `Rigidbody ballRigidBody;`
- ▶ `ballRigidbody = GetComponent<RigidBody>()`
- ▶ And then we will be able to apply forces to that rigid body which will move the ball.

Scripts

- ▶ Apply a force to the rigid body with `AddForce(Vector3)`;
- ▶ Press Play and see it go!

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour {
5     Vector3 pos;
6     Rigidbody ballRigidBody;
7     // Use this for initialization
8     void Start () {
9         ballRigidBody = GetComponent<Rigidbody> ();
10    }
11    // Update is called once per frame
12    void Update () {
13        pos = gameObject.transform.position;
14        ballRigidBody.AddForce(new Vector3(5,0,0));
15    }
16 }
```

Scripts

- ▶ Now we will add controls
- ▶ We will use the flags of the form `Input.GetKey(KeyCode.W)`

This will be true or false for example if the W key is being pressed down, then the flag will remain true.

There is also `GetKeyDown` which will detect it being pressed down and `GetKeyUp` which will detect it being released.

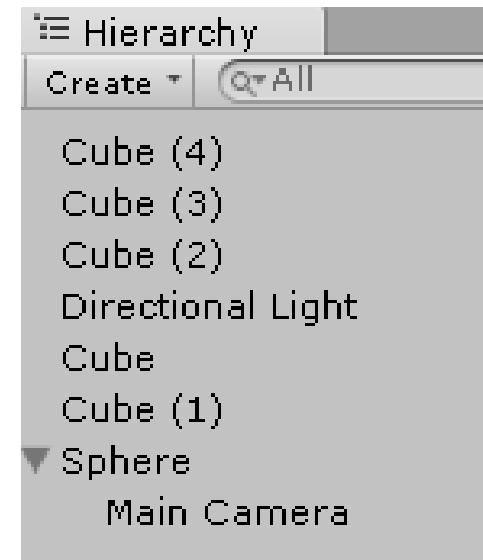
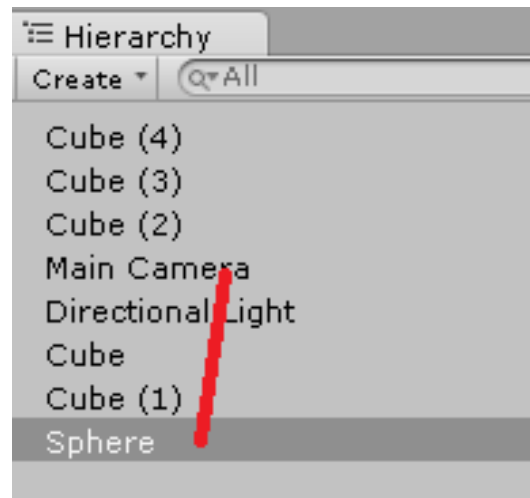
Scripts

- ▶ Use the code to the right and press play.
- ▶ You should now be able to control the ball in the scene!

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour {
5     Vector3 pos;
6     float thrust;
7     Rigidbody ballRigidBody;
8     // Use this for initialization
9     void Start () {
10         ballRigidBody = GetComponent<Rigidbody> ();
11         thrust = 5;
12     }
13     // Update is called once per frame
14     void Update () {
15         pos = gameObject.transform.position;
16         if (Input.GetKey(KeyCode.W)) {
17             ballRigidBody.AddForce(new Vector3(0,0,thrust));
18         }
19         if (Input.GetKey(KeyCode.A)) {
20             ballRigidBody.AddForce(new Vector3(-thrust,0,0));
21         }
22         if (Input.GetKey(KeyCode.S)) {
23             ballRigidBody.AddForce(new Vector3(0,0,-thrust));
24         }
25         if (Input.GetKey(KeyCode.D)) {
26             ballRigidBody.AddForce(new Vector3(thrust,0,0));
27         }
28     }
29 }
```

We can move! Its kinda like a game now!

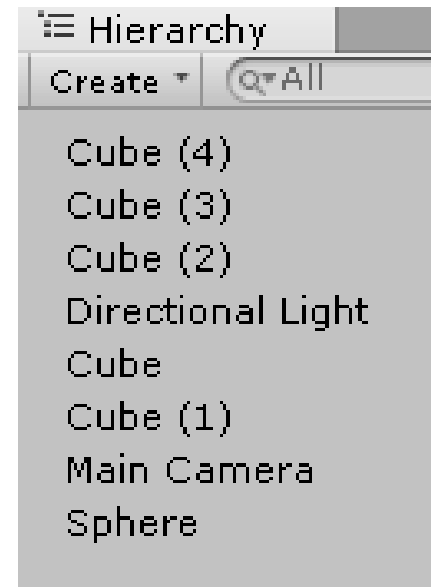
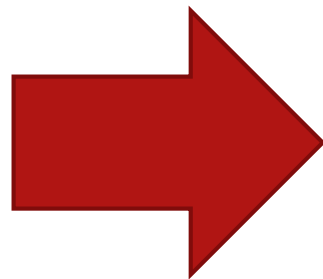
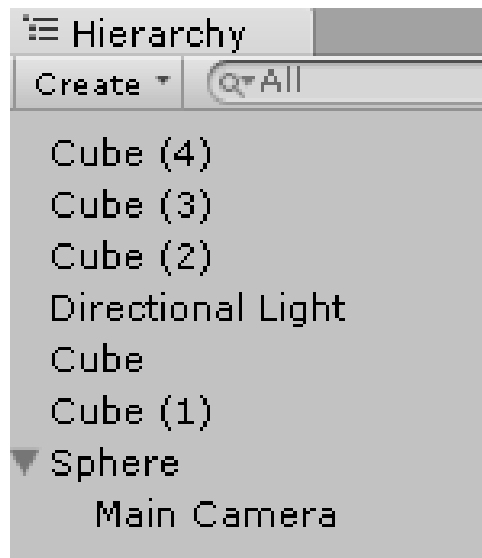
- ▶ Next, lets make the camera follow the player.
- ▶ In the Hierarchy, drag and drop the main camera on top of the sphere.



- ▶ Then press play/roll around!

Camera following

- ▶ As you can see, the camera is now rotating as if the sphere's axes are the world axes. It's rotation and position are relative to the sphere.
- ▶ This isn't very useful but it was worth a shot!
- ▶ Move the camera back off of the sphere... by clicking and dragging.



Camera follow

- ▶ Instead we will create a script on the camera that will update the camera's x and z position depending on the position of the player.
- ▶ Here we will need to access the Transform of a game object that our script is not attached to. We can do this by the function `GameObject.Find(name)`
- ▶ Select the camera, add component → New Script → camerafollow c# add
- ▶ Open the script in your text editor.

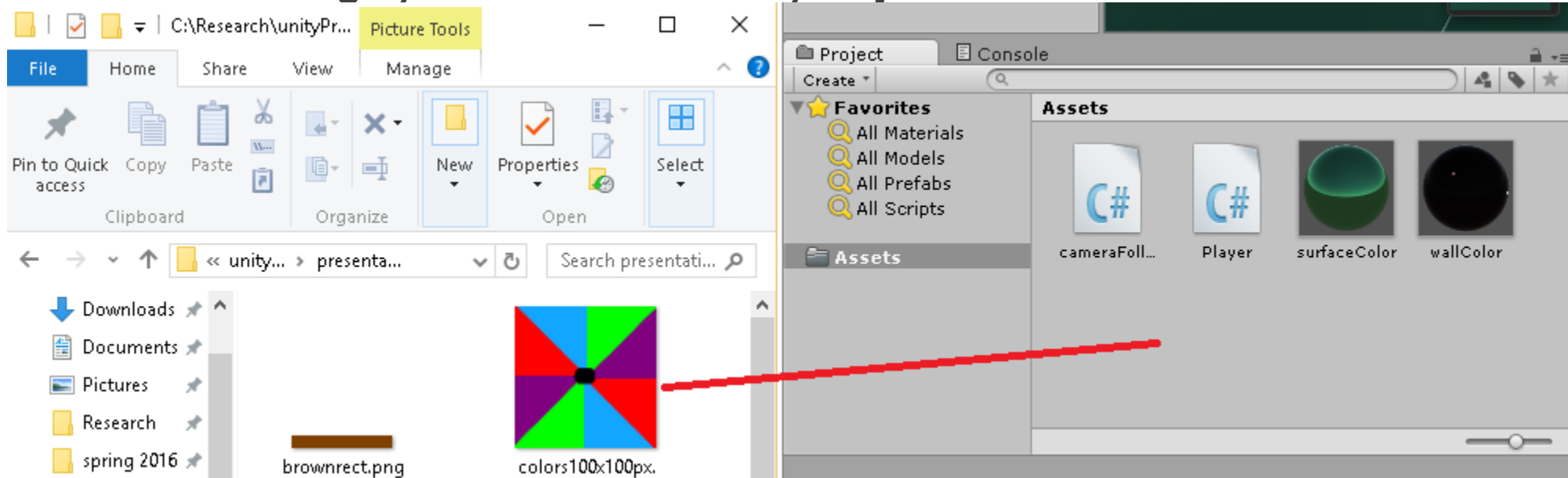
More scripts

- ▶ We will obtain the position of the player, and we will assign the camera's position to a position depending on the player's x and z coordinates.
- ▶ The script to the left will position the camera relative to the player.
- ▶ Press play and check it out!

```
cameraFollow ▶ Start ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class cameraFollow : MonoBehaviour {
5     float zOffset;
6     float cameraHeight;
7     float xOffset;
8     GameObject player;
9     Vector3 cameraPos, playerPos;
10    // Use this for initialization
11    void Start () {
12        zOffset = -10;
13        cameraHeight = 4;
14        xOffset = 0;
15        player = GameObject.Find ("Sphere");
16    }
17    // Update is called once per frame
18    void Update () {
19        playerPos = player.transform.position;
20        cameraPos = gameObject.transform.position;
21        cameraPos = new Vector3 (playerPos.x + xOffset,
22                                cameraHeight, playerPos.z + zOffset);
23        gameObject.transform.position = cameraPos;
24    }
25 }
26
```

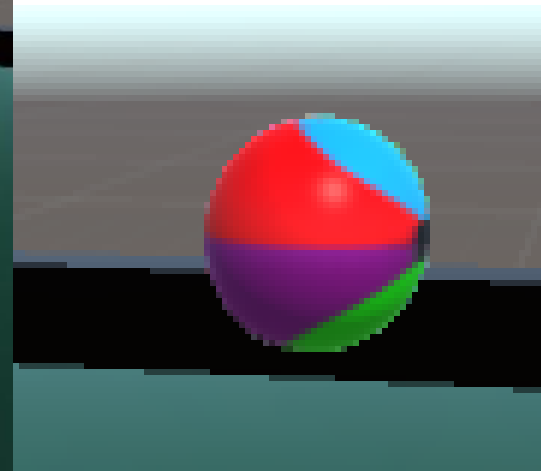
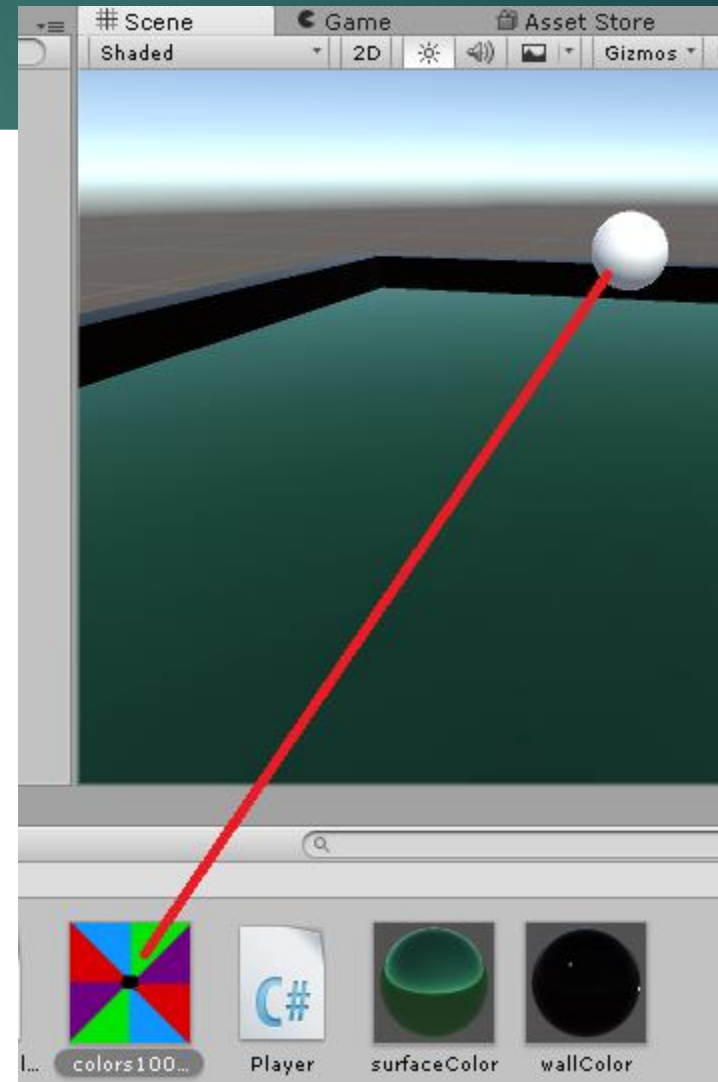
Add color to the player

- ▶ The ball is rotating even if it doesn't seem like it. Here we will add an image to be the material for the ball.
- ▶ Drag and drop the colors100x100.png image in the assets folder or any other image you want into the Unity Projects window:



Adding color to the player.

- ▶ Next Drag the image into the scene window and drop it on top of the sphere. This will automatically create a new material and apply it to the sphere! The material will have the image attached to it. (you might need to move the directional light away from your ball.)
- ▶ This is a good example of how images might wrap on 3D objects in ways not immediately obvious. (kinda important to know about if you have complex models like people)

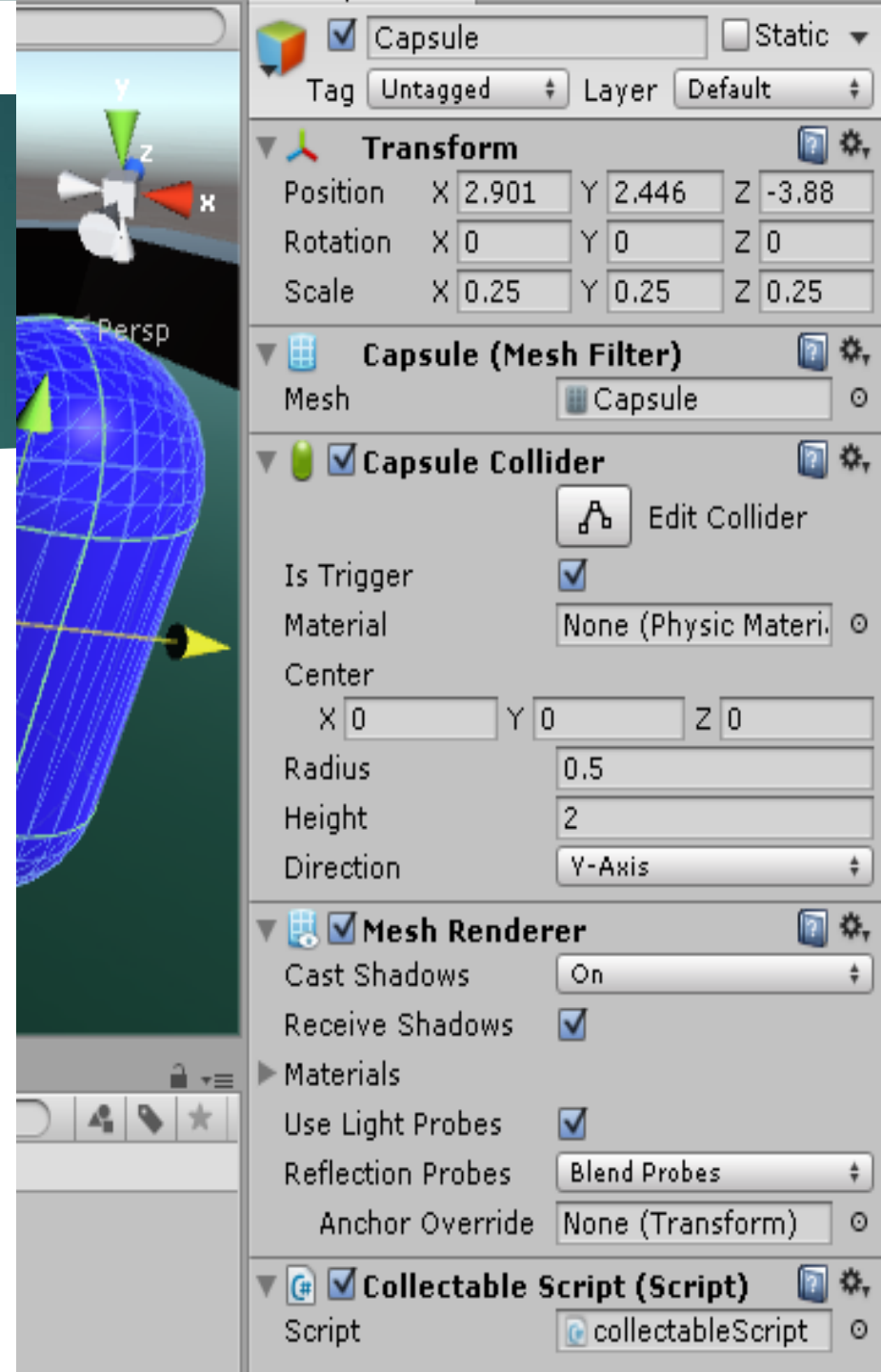


Now to add collectables

- ▶ So now we can roll around our arena and run into walls with our colorful ball.
- ▶ Next we will create objects that we will place around the arena. When the player runs over these objects they will be destroyed and the player will grow in size.
- ▶ First, create a new 3D object in the scene like before
Game Objects → 3D Object → capsule is what we will use.

Creating a collectable

- ▶ After creating the new object, create a new material for it and apply the material so that it has an interesting color.
- ▶ Next, scale the object to (0.25,0.25,0.25)
- ▶ Then, under the inspector → capsule collider, check “Is Trigger”
This will make the collider non-solid so that things can pass through it. However we will be able to detect in scripts when colliders intersect the trigger.
- ▶ Finally Create a script for the object, something like “collectableScript”



Collectable behavior

- ▶ I ran out of time creating slides :((((5:50pm) but will keep going