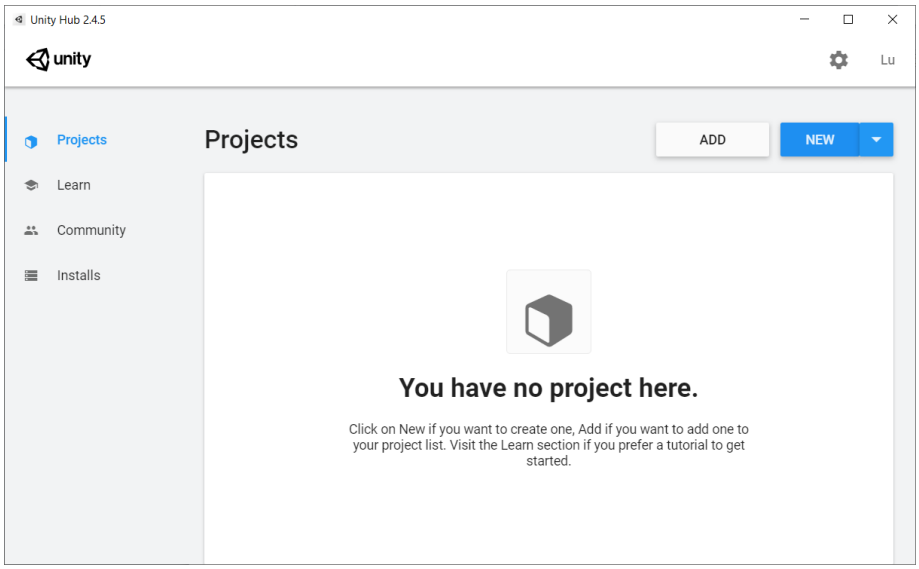
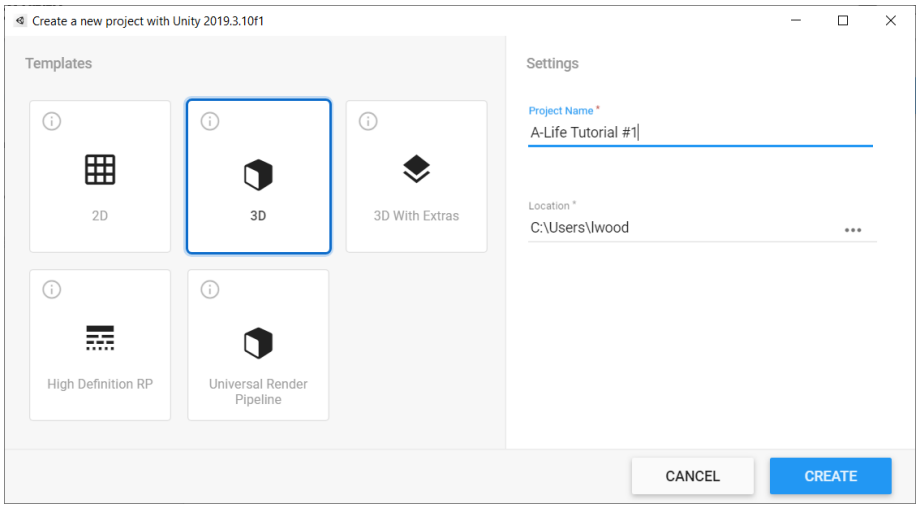
Step 1 – Open Unity and create a project

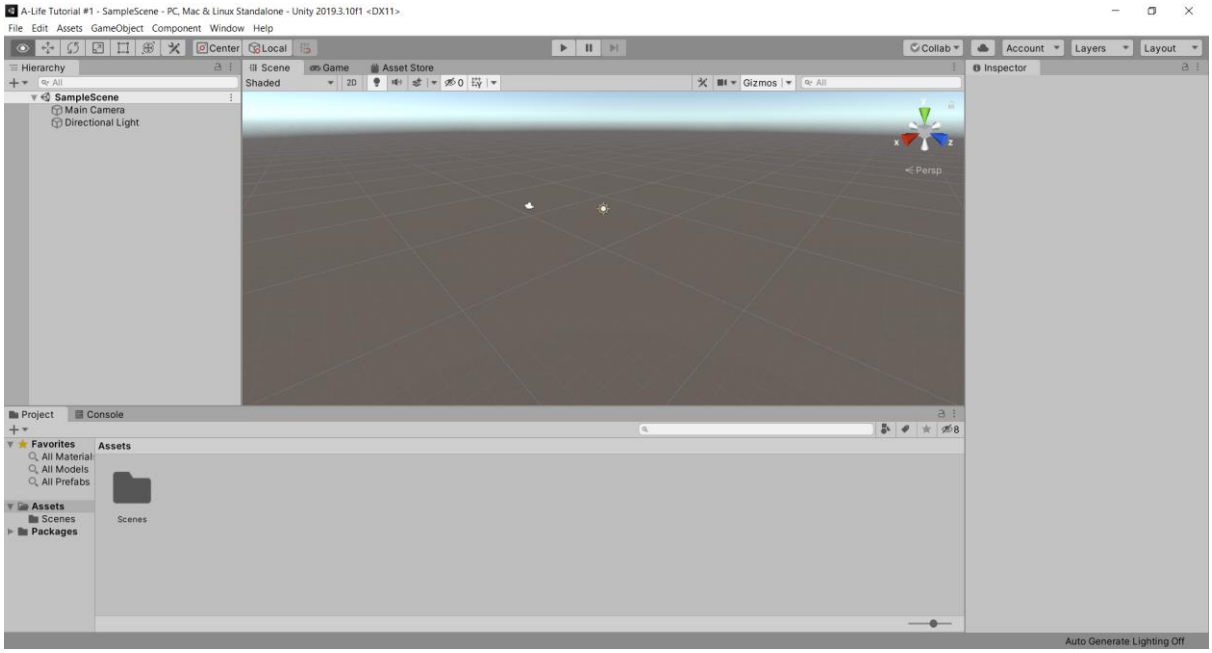
You will be presented with the following screen. Select new from the top right of the window.



Create a project with the following details:

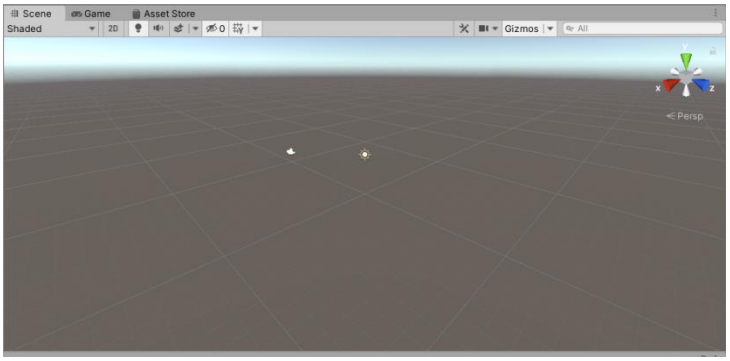


Step 2 – A tour of the interface



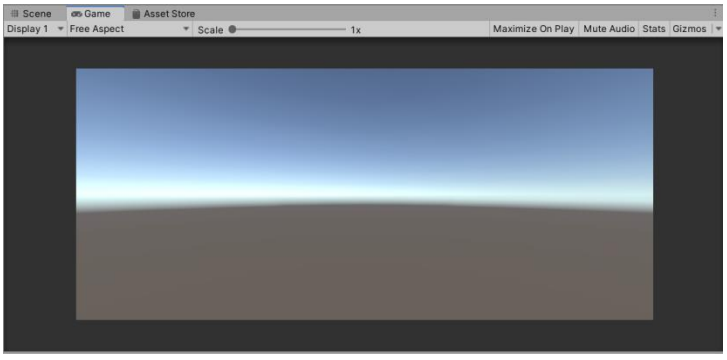
Scene view

overall view, everything in the world. The only objects you will see are the camera and light source.



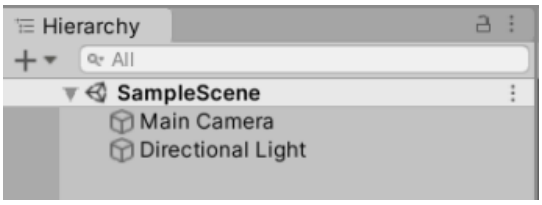
Game window

Specific perspective from a camera (what the human can see when playing)



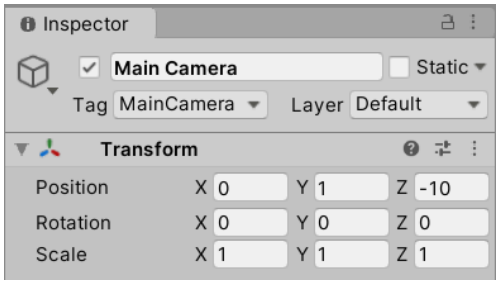
Hierarchy

List of items in your world, by default starts with the camera and light source. You can click on items to select them and this will also show their properties in the inspector on the right hand side of the screen. The game will render from the top down so you may need to consider this in your development (the item at the bottom will be rendered on the top).



Inspector

Properties for the selected item in the Hierarchy



Toolbar

Move/edit GameObjects

Project window

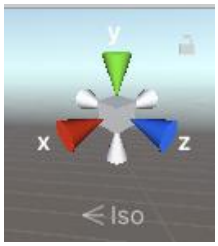
All the assets that are in you project are in here

Runtime controls

Play the game (run the world)

View Cube

Change orientation using cube



Movement shortcuts

For extra efficiency, all of these controls can also be used regardless of which transform tool is selected. The most convenient controls depend on which mouse or track-pad you are using:

Action	3-button mouse	2-button mouse or track-pad	Mac with only one mouse button or track-pad
Move	Hold **Alt**+middle mouse button, then drag	Hold **Alt**+**Control**+left-click, then drag	Hold **Alt**+**Command**+left-click, then drag
Orbit (Not available in 2D mode)	Hold **Alt**+left-click, then drag	Hold **Alt**+left-click, then drag	Hold **Alt**+left-click, then drag
Zoom	Use the scroll wheel, or hold **Alt**+right-click, then drag	Hold **Alt**+right-click, then drag	Use the two-finger swipe method to scroll in and out, or hold **Alt**+**Control**+left-click, then drag
Change speed (only available in Flythrough mode)	Use the scroll wheel while moving.	Drag with two fingers while moving.	Drag with two fingers while moving.

Step 3 – GameObjects

GameObjects are the fundamental objects in Unity that represent characters, props and scenery. They do not accomplish much in themselves but they act as containers for Components, which implement the real functionality.

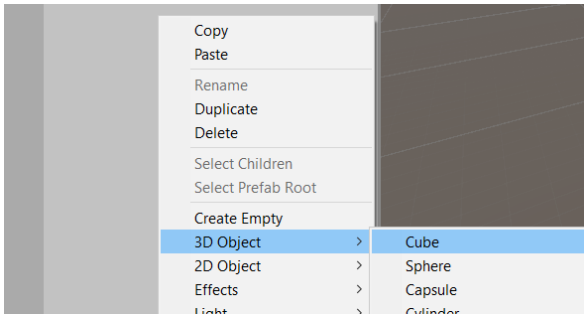
Pre-made GameObjects

You will notice 2 pre-made GameObjects when you start a project, these are:

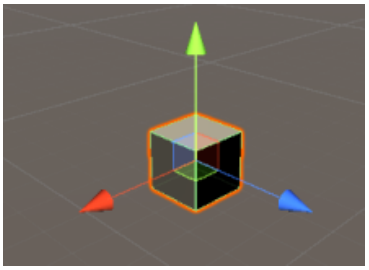
- Camera – Which displays what the human player will see when the game is running
- Directional light – Which provides overall lighting for the scene

Creating basic GameObjects

Right click in the hierarchy window then you will see a list of the GameObjects you can select.

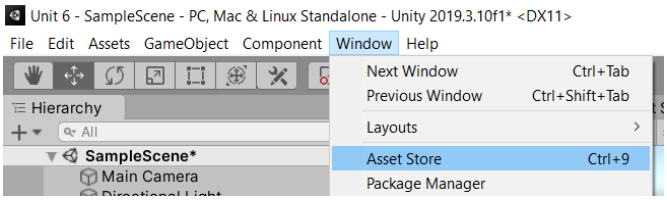


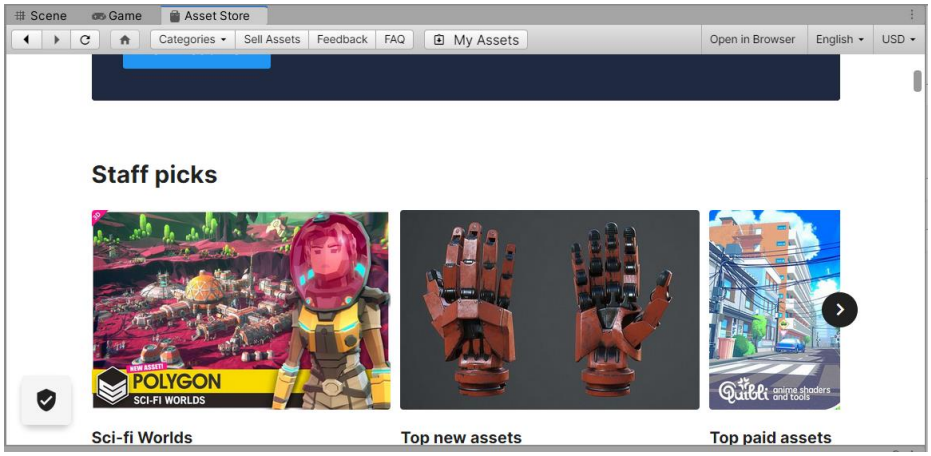
For this lesson select Cube, you will see the item appear in the scene view.



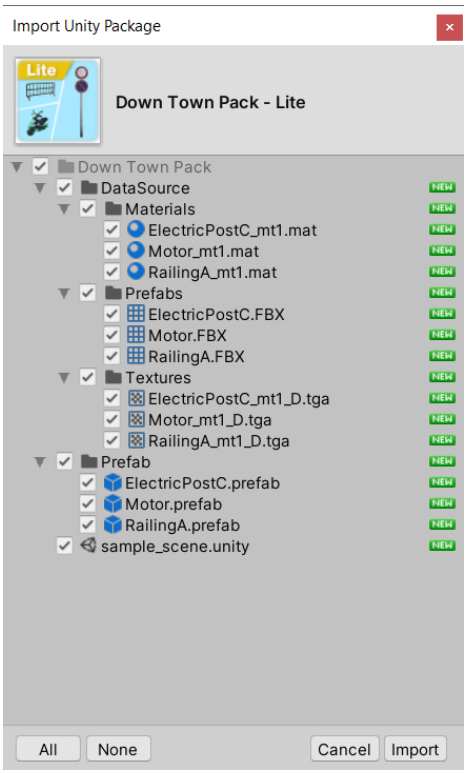
Importing GameObjects from the asset store

For rapid development you can download GameObjects from the asset store, there are both free and paid assets available for your projects.



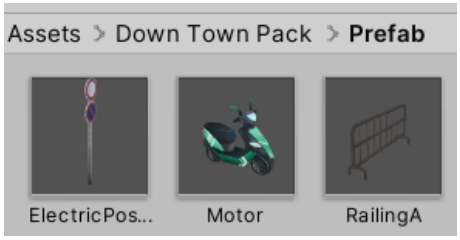


Search for “Down Town Pack – Lite” then select and download the asset. The download button will be replaced with an import button, select import and the following screen will appear:



Again select import and once imported there will be an additional folder in your asset list

Enter the folder and select prefabs to see assets



To add an asset to the world simply drag and drop onto the scene.

Pre-fabs

Unity’s Prefab system allows you to create, configure, and store a GameObject complete with all its components, property values, and child GameObjects as a reusable Asset. The Prefab Asset acts as a template from which you can create new Prefab instances in the Scene

When you want to reuse a GameObject configured in a particular way – like a non-player character (NPC), prop or piece of scenery – in multiple places in your Scene, or across multiple Scenes in your Project, you should convert it to a Prefab. This is better than simply copying and pasting the GameObject, because the Prefab system allows you to automatically keep all the copies in sync.

Any edits that you make to a Prefab Asset are automatically reflected in the instances of that Prefab, allowing you to easily make broad changes across your whole Project without having to repeatedly make the same edit to every copy of the Asset.

You can nest Prefabs inside other Prefabs to create complex hierarchies of objects that are easy to edit at multiple levels.

However, this does not mean all Prefab instances have to be identical. You can override settings on individual prefab instances if you want some instances of a Prefab to differ from others. You can also create variants of Prefabs which allow you to group a set of overrides together into a meaningful variation of a Prefab.

Step 4 – Modifying GameObjects

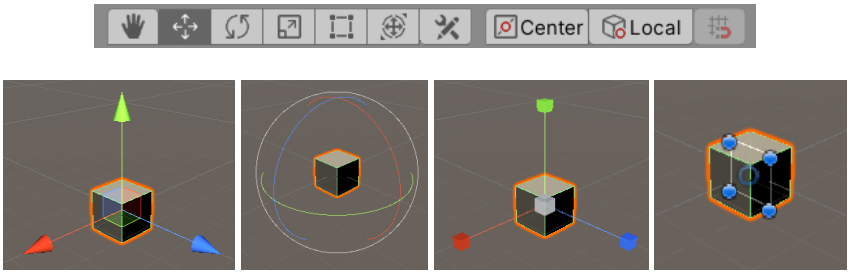
Click on the cube you created and we will explore the different ways of modifying the object.

Transform

The Transform is used to store a GameObject's position, rotation, scale and parenting state and is thus very important. A GameObject will always have a Transform component attached - it is not possible to remove a Transform or to create a GameObject without one.

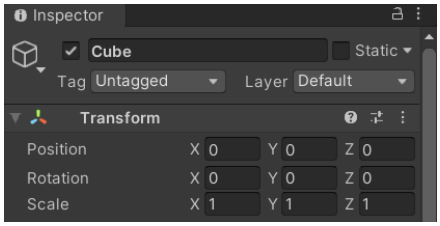
Toolbar to transform GameObject

Use the tools in the toolbar to modify GameObject's in the scene. Click on the coloured lines and points to edit the GameObject using these tools



Inspector to transform GameObject

Values can be entered directly into the inspector to transform a GameObject.



Code to transform GameObject

All of the transform properties of GameObject's can be accessed/modified in code, see the following link for more details of this, we will be covering this in future sessions:

https://docs.unity3d.com/ScriptReference/Transform.html

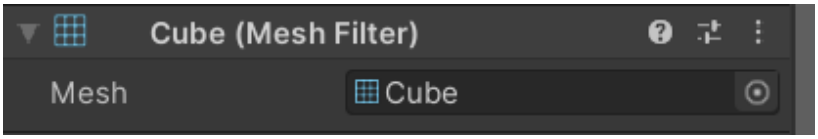
Good practice regarding transforms for GameObject's

Ideally, you should not adjust the **Scale** of your object in the Transform Component. The best option is to create your models at real-life scale so you won't have to change your Transform's scale. The next best option is to adjust the scale at which your mesh is imported in the **Import Settings** for your individual mesh. Certain optimizations occur based on the import size, and instantiating an object that has an adjusted scale value can decrease performance. For more information, see the section about optimizing scale on the Rigidbody component reference page.

When parenting Transforms, it is useful to set the parent's location to <0,0,0> before adding the child. This means that the local coordinates for the child will be the same as global coordinates making it easier to be sure you have the child in the right position.

Mesh Filter

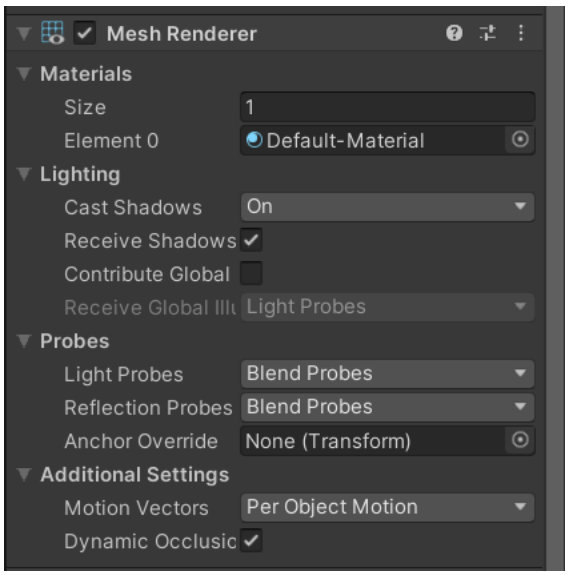
When importing mesh assets, Unity automatically creates a Skinned Mesh Renderer if the mesh is skinned, or a Mesh Filter along with a Mesh Renderer, if it is not.



In order for non-skinned Meshes to be rendered, both the Mesh Filter and Mesh Renderer components must be present for legacy reasons.

Mesh renderer

The Mesh Renderer takes the geometry from the Mesh Filter and renders it at the position defined by he GameObject's Transform component.



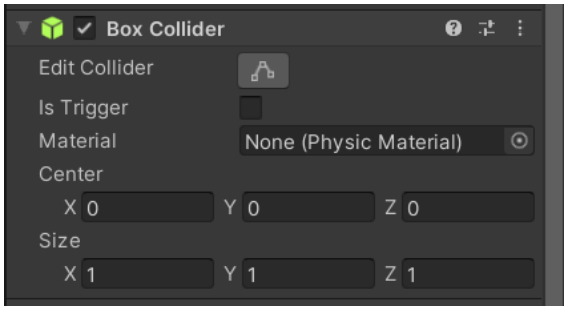
Size - Specify the number of Materials in the Mesh Renderer. If you decrease the size of the list of Materials, Unity deletes the elements at the end of the list.

Element - A list of the Materials in the Mesh Renderer, in numeric order. The first element is always named Element 0.

Full details can be found via the lin below:
https://docs.unity3d.com/Manual/class-MeshRenderer.html

Collider

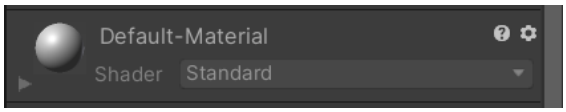
Collider components define the shape of a GameObject for the purposes of physical collisions. A collider, which is invisible, does not need to be the exact same shape as the GameObject's mesh. A rough approximation of the mesh is often more efficient and indistinguishable in gameplay.



The simplest (and least processor-intensive) colliders are primitive collider types. In 3D, these are the Box Collider, Sphere Collider and Capsule Collider. In 2D, you can use the Box Collider 2D and Circle Collider 2D. You can add any number of these to a single GameObject to create compound colliders.

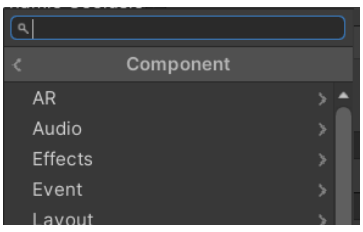
Material

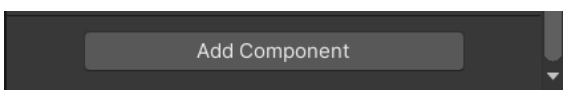
In Unity (and in many 3D modelling aspects), a Material is a file that contains information about the lighting of an object with that material. Notice how a gray sphere denotes the material, with some light coming in from the top.



More components

There are numerous additional components that can be added to game objects, we will only cover a small number in this module to achieve what we need.

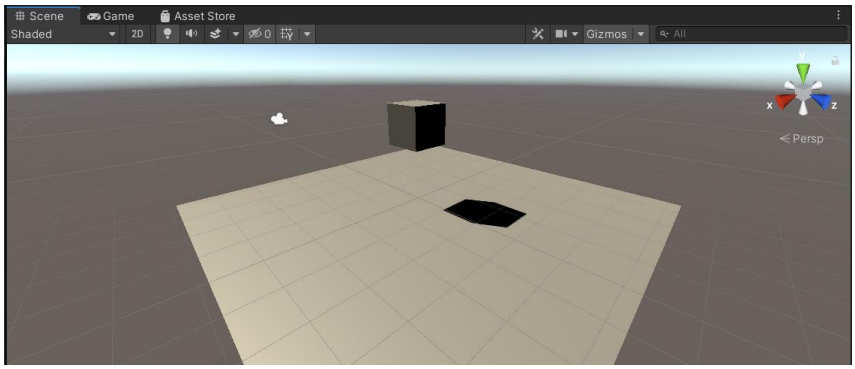




Step 5 – Lets start creating our world

Add a plane

You should already have a cube, now create a plane in the hierarch. The position of the plane should be 0,0,0. Move the cube to 0,2,0 so it looks like the image below

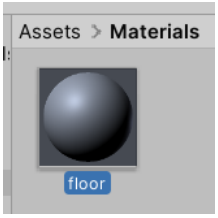


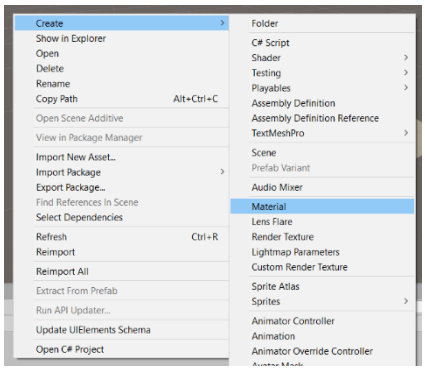
Organise your assets

Add a new folder called materials

Create a material

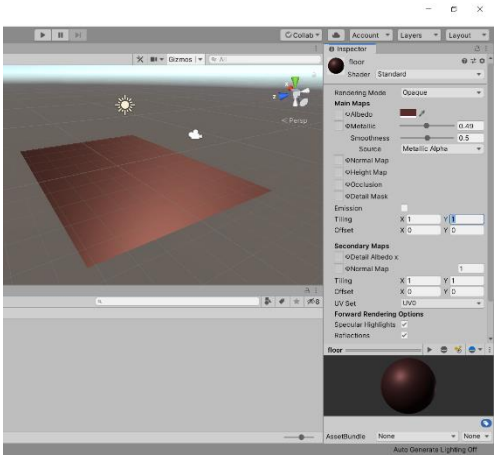
Now create a material called floor





Edit the materal

Edit the properties of the material in the inspector then apply the material by dragging it onto the GameObject.



Cube material

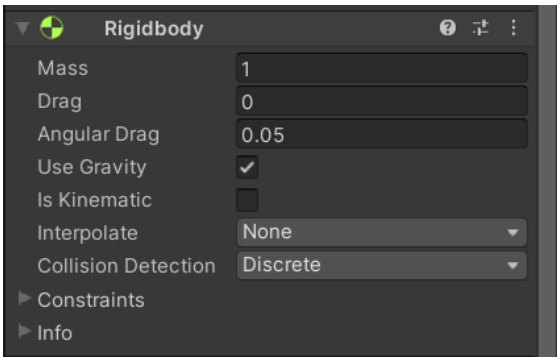
Repeat this process and make a material then apply it to the cube

Applying physics to our world

Unity helps you simulate physics in your Project to ensure that the objects correctly accelerate and respond to collisions, gravity, and various other forces. Unity provides different physics engine implementations which you can use according to your Project needs: 3D, 2D, object-oriented, or data-oriented. We will now implement some basic physics.

Add Rigidbody to the cube

Select the cube and add a Rigidbody component, the following item will be added to the inspector



Ensure gravity is ticked then run the game to see what happens.

The cube should fall and land on the plane

Repeat the process with a capsule

Try repeating this process with a capsule setting the angle to 10 degrees on 1 axis an see what happens.

You should notice that the capsule falls on its side, this shows how the physics engine is working to the shape of the objects. To stop this you can constrain some of the axis in the Rigidbody component.

Adding a script

Create a script called player, enter the code below then add this component to the cube.

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Player : MonoBehaviour
{
    private float horizontalInput;
    private Rigidbody rigidBodyComponent;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        rigidBodyComponent = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space) == true)
        {
            rigidBodyComponent.AddForce(Vector3.up * 5, ForceMode.VelocityChange);
        }
        horizontalInput = Input.GetAxis("Horizontal");
    }

    Unity Message | 0 references
    void FixedUpdate()
    {
        rigidBodyComponent.velocity = new Vector3(horizontalInput, rigidBodyComponent.velocity.y, 0);

    }
}
```