Unity and GitHub

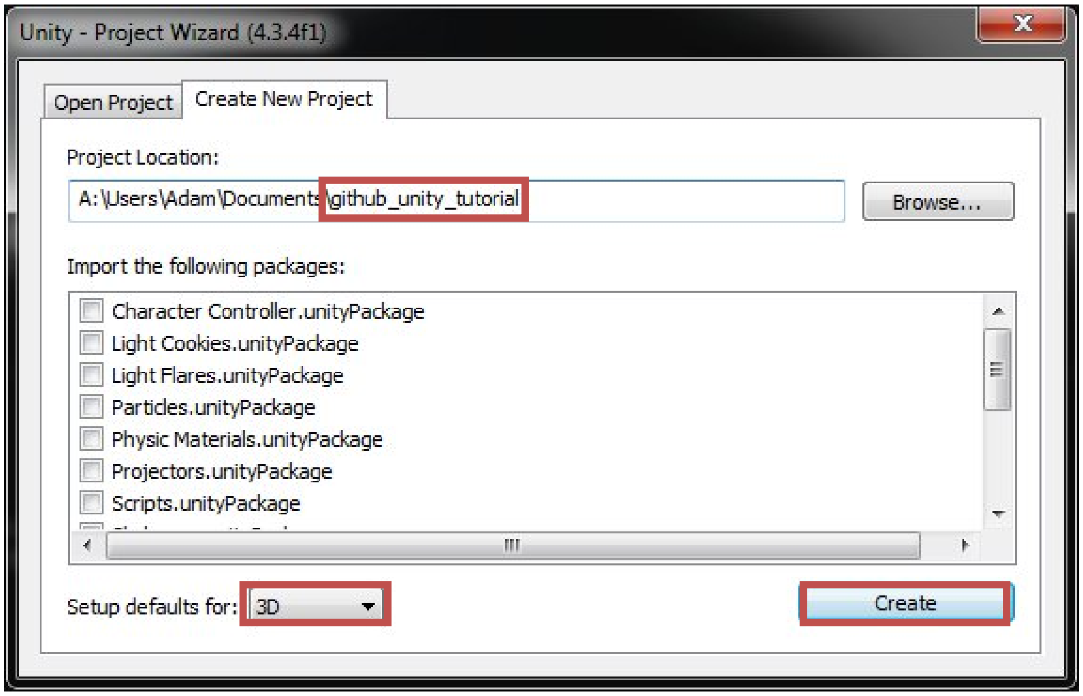
This tutorial is going to go through some of the basics of using unity and adding the project to git hub. Before we start you're going to need to make sure that Unity and the GitHub GUI are installed and that you have a GitHub account, if you don't you can make a free account here : https://github.com/

Unity is a 3D games engine that uses Physx, It allows for JavaScript and C#. It also makes use of 3D objects and can import then directly from 3Ds max.

GitHub is a Git repository web-based hosting service which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. (wikipedia)

More GitHub tutorials at https://try.github.com/

The first thing to do is to open unity and click on create a new project and give it a name in the project location box, you can set the defaults for a 2D or 3D project (though this tutorial will use the 3D defaults) and then click create like below.



Unity - Project Wizard (4.3.4f1)

Open Project | Create New Project

Project Location:

A:\Users\Adam\Documents\github_unity_tutorial

Browse...

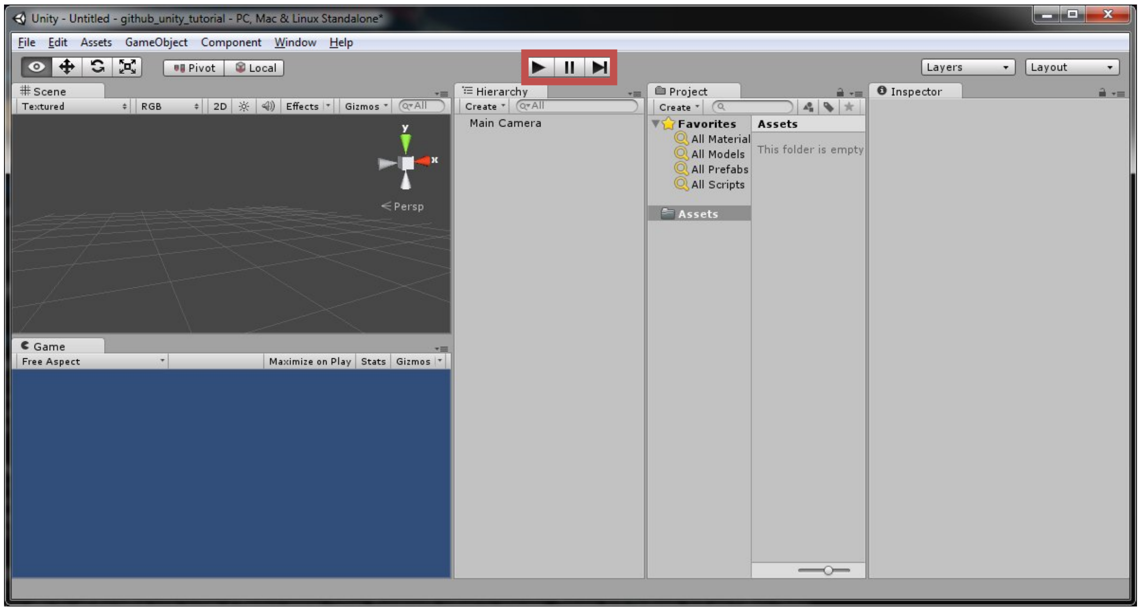
Import the following packages:

- [ ] Character Controller.unityPackage
- [ ] Light Cookies.unityPackage
- [ ] Light Flares.unityPackage
- [ ] Particles.unityPackage
- [ ] Physic Materials.unityPackage
- [ ] Projectors.unityPackage
- [ ] Scripts.unityPackage

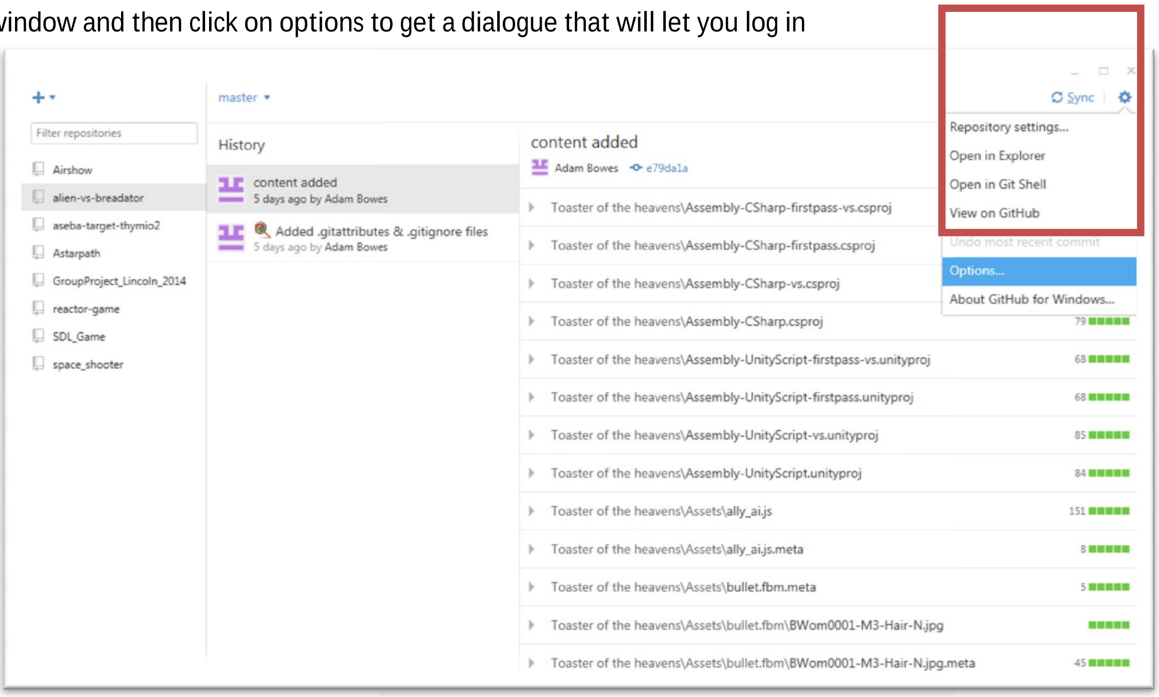
Setup defaults for: 3D

Create

You should then find yourself with the unity project open and ready to go. At the top there are 3 buttons to run your project, pause it and stop. On the right there are a few tabs which show the objects in the scene, the folders and files in the project and an inspector. If your screen doesn't look like the one below then you might want to click on window > layouts > 2 by 3.



Now let's add the project to git hub. open the GitHub GUI and click on the gear icon to open a window and then click on options to get a dialogue that will let you log in



Airshow
alien-vs-breadator
aseba-target-thymio2
Astarpath
GroupProject_Lincoln_2014
reactor-game
SDL_Game
space_shooter

History

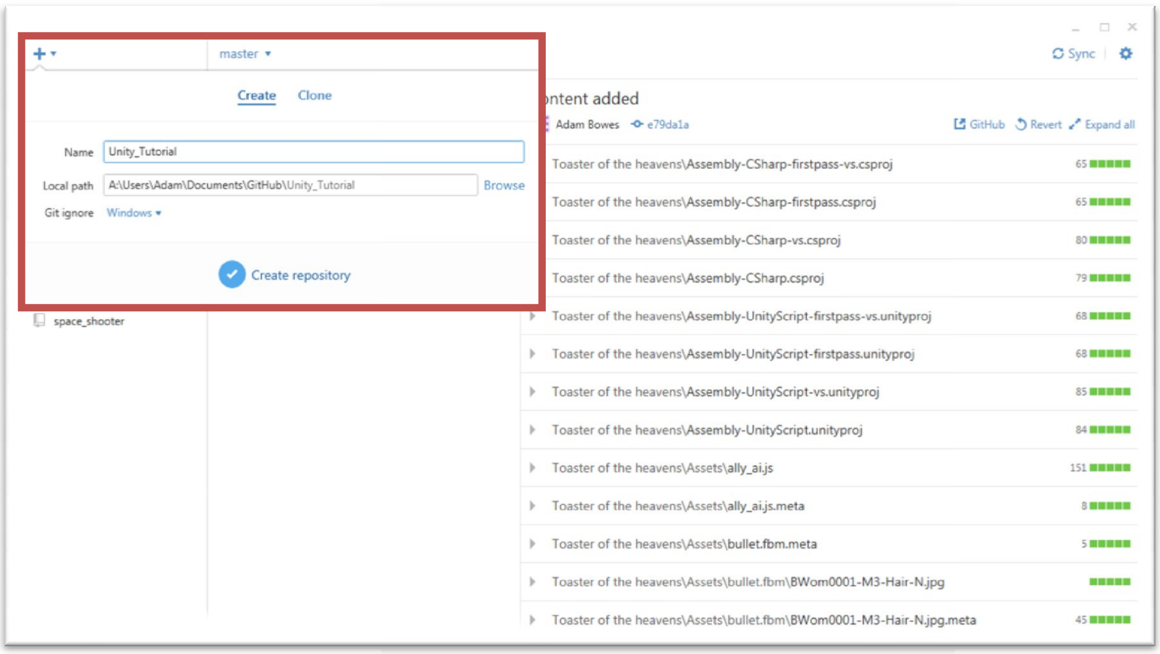
content added
5 days ago by Adam Bowes

Added .gitattributes & .gitignore files
5 days ago by Adam Bowes

content added
Adam Bowes e79da1a

Toaster of the heavens\Assembly-CSharp-firstpass-vs.csproj	
Toaster of the heavens\Assembly-CSharp-firstpass.csproj	
Toaster of the heavens\Assembly-CSharp-vs.csproj	
Toaster of the heavens\Assembly-CSharp.csproj	79
Toaster of the heavens\Assembly-UnityScript-firstpass-vs.unityproj	68
Toaster of the heavens\Assembly-UnityScript-firstpass.unityproj	68
Toaster of the heavens\Assembly-UnityScript-vs.unityproj	85
Toaster of the heavens\Assembly-UnityScript.unityproj	84
Toaster of the heavens\Assets\ally_ai.js	151
Toaster of the heavens\Assets\ally_ai.js.meta	8
Toaster of the heavens\Assets\bullet.fbm.meta	5
Toaster of the heavens\Assets\bullet.fbm\BWom0001-M3-Hair-N.jpg	
Toaster of the heavens\Assets\bullet.fbm\BWom0001-M3-Hair-N.jpg.meta	45

. After logging in, click on the + icon to create a new repository for your unity project, pick a suitable name and click create, you can leave the path as it is.



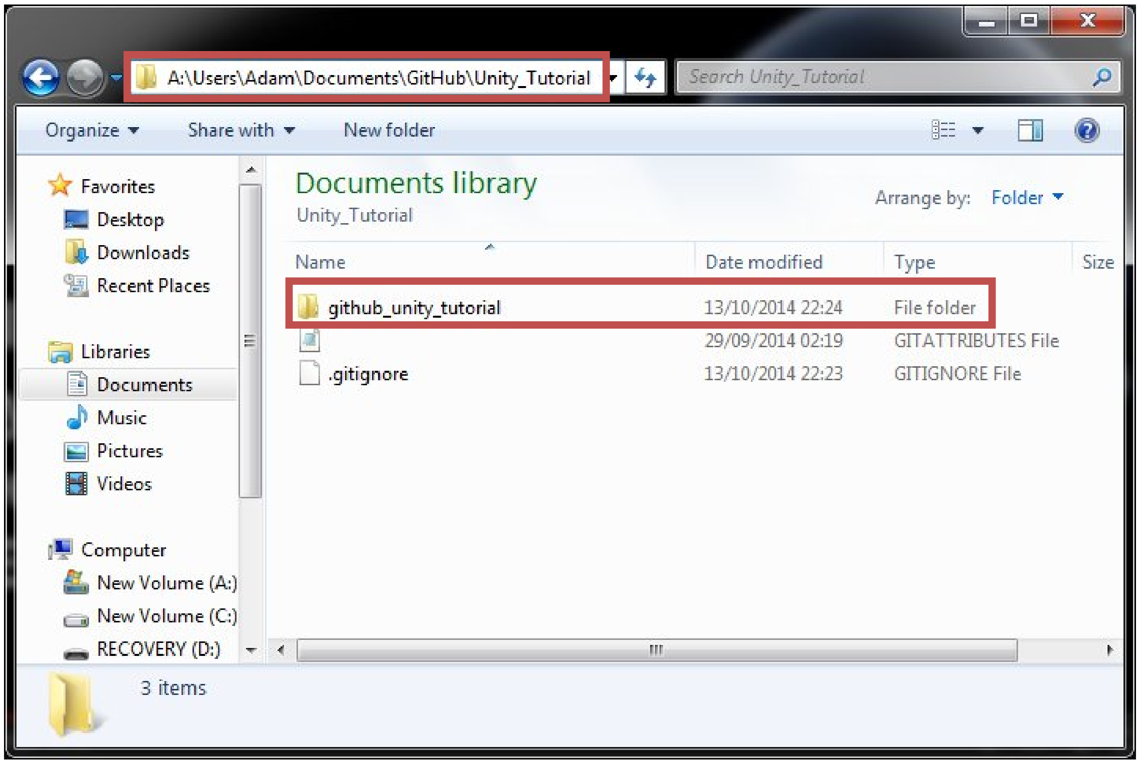
Name: Unity_Tutorial
Local path: A:\Users\Adam\Documents\GitHub\Unity_Tutorial
Git ignore: Windows

space_shooter

content added
Adam Bowes e79da1a

Toaster of the heavens\Assembly-CSharp-firstpass-vs.csproj	65
Toaster of the heavens\Assembly-CSharp-firstpass.csproj	65
Toaster of the heavens\Assembly-CSharp-vs.csproj	80
Toaster of the heavens\Assembly-CSharp.csproj	79
Toaster of the heavens\Assembly-UnityScript-firstpass-vs.unityproj	68
Toaster of the heavens\Assembly-UnityScript-firstpass.unityproj	68
Toaster of the heavens\Assembly-UnityScript-vs.unityproj	85
Toaster of the heavens\Assembly-UnityScript.unityproj	84
Toaster of the heavens\Assets\ally_ai.js	151
Toaster of the heavens\Assets\ally_ai.js.meta	8
Toaster of the heavens\Assets\bullet.fbm.meta	5
Toaster of the heavens\Assets\bullet.fbm\BWom0001-M3-Hair-N.jpg	
Toaster of the heavens\Assets\bullet.fbm\BWom0001-M3-Hair-N.jpg.meta	45

Now close the unity project, find the folder and copy it in to the GitHub folder, the addresses will be similar to the ones below. Unity should default to documents folder for saving and in the documents folder there should also be a GitHub folder.



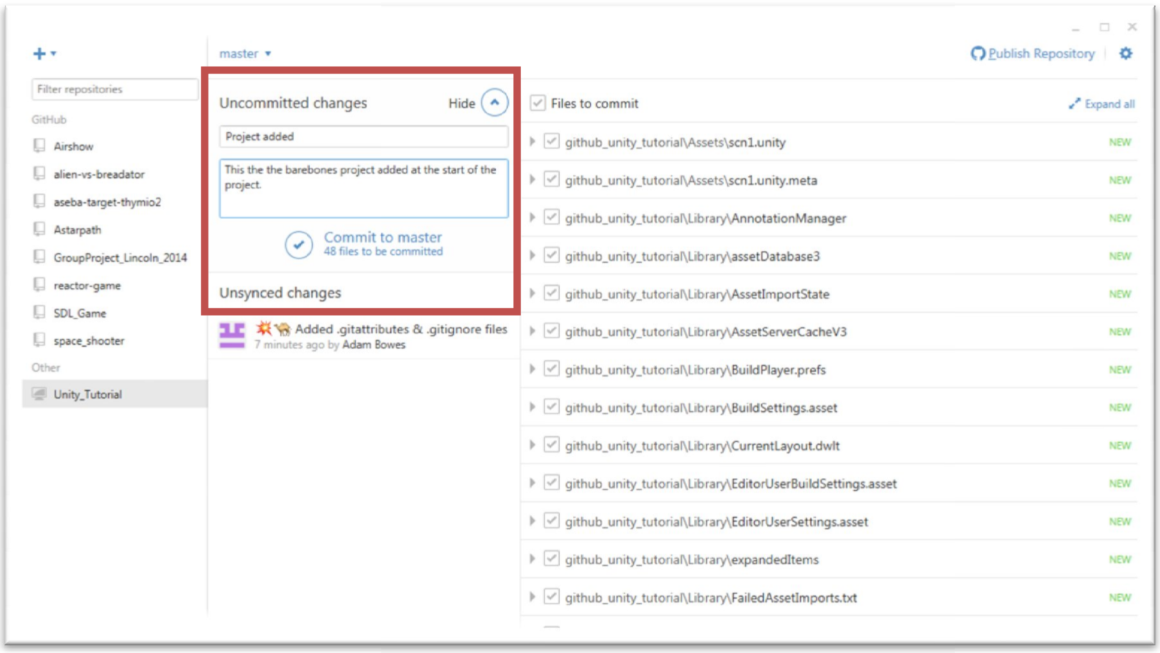
A:\Users\Adam\Documents\GitHub\Unity_Tutorial

Documents library
Unity_Tutorial

Name	Date modified	Type	Size
github_unity_tutorial	13/10/2014 22:24	File folder	
	29/09/2014 02:19	GITATTRIBUTES File	
.gitignore	13/10/2014 22:23	GITIGNORE File	

3 items

GitHub should now have tab saying "Uncommitted changes". Clicking on the show icon will show the files to be added and let you enter the summary and description for this commit. Commits are used for version control in case some change breaks the project, you can revert to a previous version.



Uncommitted changes

Project added

This the the barebones project added at the start of the project.

Commit to master
48 files to be committed

Unsynced changes

Added .gitattributes & .gitignore files
7 minutes ago by Adam Bowes

Files to commit

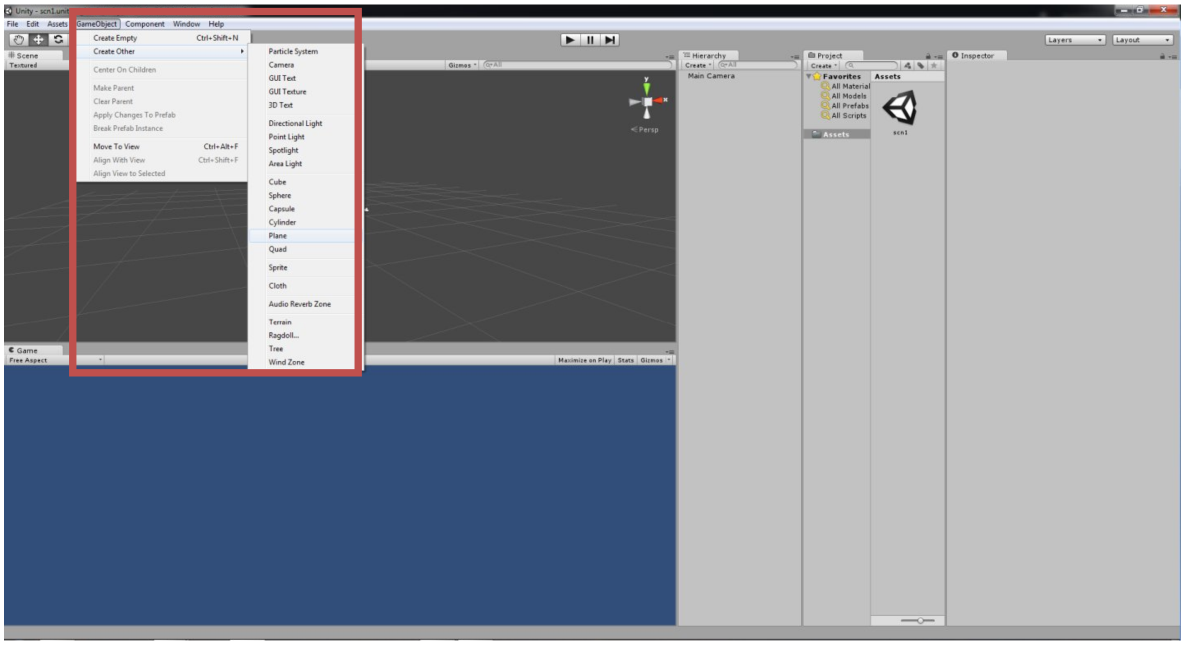
github_unity_tutorial\Assets\scn1.unity	NEW
github_unity_tutorial\Assets\scn1.unity.meta	NEW
github_unity_tutorial\Library\AnnotationManager	NEW
github_unity_tutorial\Library\assetDatabase3	NEW
github_unity_tutorial\Library\AssetImportState	NEW
github_unity_tutorial\Library\AssetServerCacheV3	NEW
github_unity_tutorial\Library\BuildPlayer.prefs	NEW
github_unity_tutorial\Library\BuildSettings.asset	NEW
github_unity_tutorial\Library\CurrentLayout.dwlt	NEW
github_unity_tutorial\Library\EditorUserBuildSettings.asset	NEW
github_unity_tutorial\Library\EditorUserSettings.asset	NEW
github_unity_tutorial\Library\expandedItems	NEW
github_unity_tutorial\Library\FailedAssetImports.txt	NEW

After Clicking commit the master branch of the repository you made will be updated to include the project you added. Clicking "Publish Repository" will add the project to GitHub so the project will be available online. after this, instead of publish it will say sync which will update the online version to match the changes you've made to the local copy when clicked. Now that git hub is set up, all you need to do in future is to commit changes and sync it, and if you go on to another computer you can log in to GitHub and download the repository and continue working on it.

Now open unity again and if your project doesn't open automatically then tell unity where to find the folder and to open it.

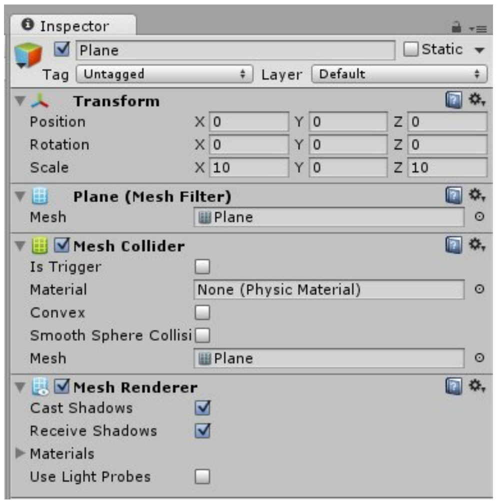
For this tutorial we will make a small physics game about knocking over towers of blocks.

Start by adding a plane and position it to 0,0,0.



Now make sure the box is selected and set its x scale to 10 and the z scale to10 so this can be our floor.

Transform defines the position, rotation and scale of the object. Below that is the mesh of the object. The mesh collider refers to the way it detects collisions, and in this case it uses the same mesh as he plane. The mesh renderer makes it so you can see the object.


Inspector

☑ Plane ☐ Static

Tag Untagged Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 10	Y 0	Z 10

Plane (Mesh Filter)

Mesh Plane

☑ **Mesh Collider**

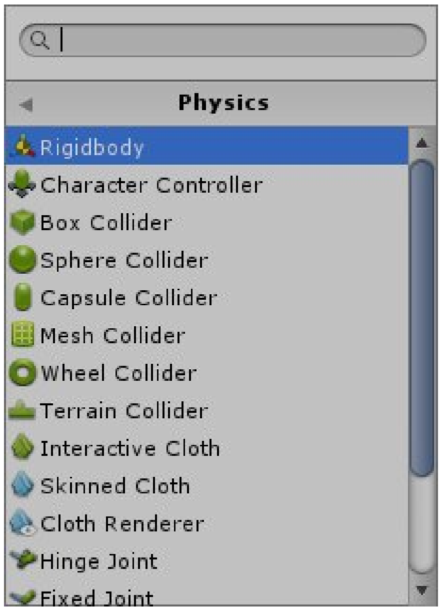
Is Trigger	☐
Material	None (Physic Material)
Convex	☐
Smooth Sphere Collisi	☐
Mesh	Plane

☑ **Mesh Renderer**

Cast Shadows	☑
Receive Shadows	☑
Materials	
Use Light Probes	☐

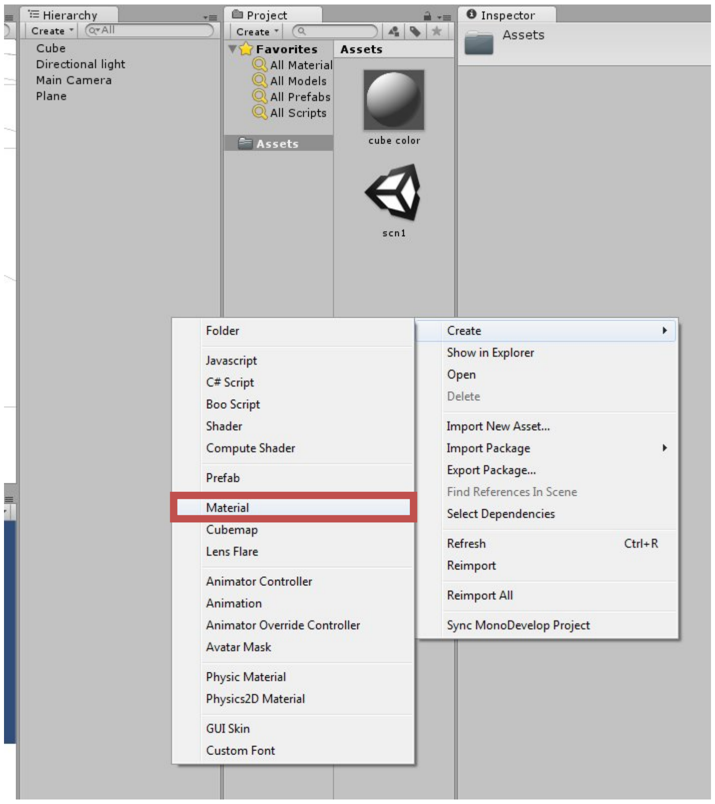
Now we will add a tower of blocks.

In the same menu where the plane object was, there is also a cube, create one and set its position to 0,2,0 this will place it above the plane. Click add component and select physics and then rigid body.



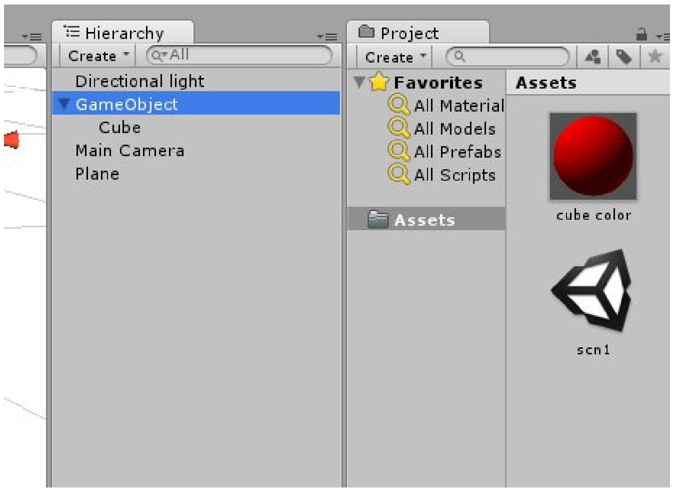
A rigid body means that now Physx will update its position, you can set its properties in the inspector under Rigidbody. If you click on the play button you should see that the game starts running and the cube drops down on to the plane and stops, clicking the play button again will stop it running. At this point you might want to add a light to make things easier to see, the directional light can be found in the same menu as the other objects. If the plane remains dark then ensure none of its scaling values are 0, if they are set them to 1. This might be a good point to go to the GitHub window and commit your changes.

To make the cube stand out we're going to give it a colour, to do this we need to right click in the assets window and create a material

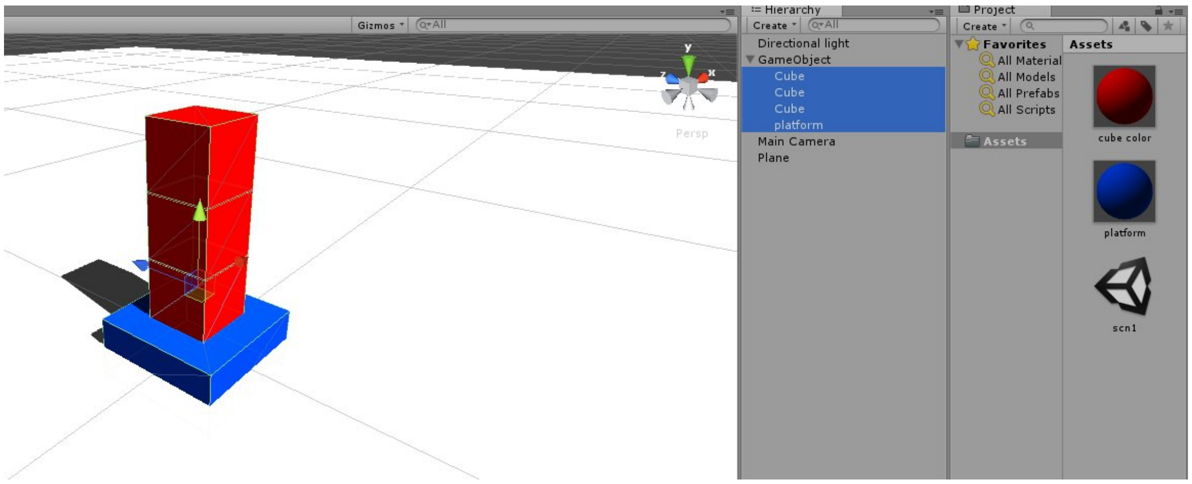


now click on the material and in the inspector set the main colour to red. Then click on the material in the assets window and drag it on to the cube in the scene window.

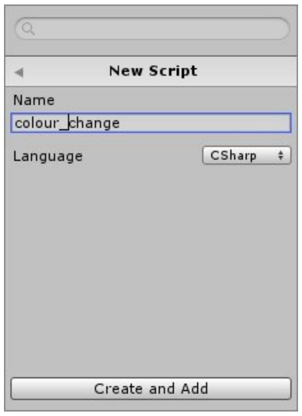
Now we have a coloured cube, lets make a tower for the player to knock down. Create an empty game object from GameObject > create empty, now set the objects position to 0,0,0. In the hierarchy window, drag the cube on to the game object so that the cube is a child of the object and it looks like below.



now, create another cube, make a new material so this cube can have a different colour (I used blue) and make its x and z scale 2. now use CTRL - C and CTRL - V to make several copies of the red cube and use either the mouse or the inspector to stack them on top of the blue cube. Ypu should end up with something similar to screenshot below, remember to make sure all of the cubes are in the gameobject holder.

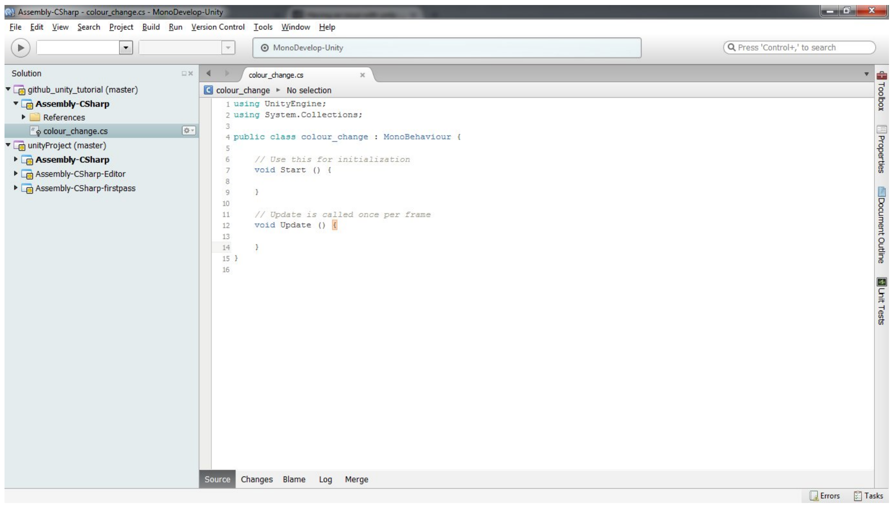


Let's make it so that if one of the red cubes hits the plane it goes green. Click on a cube and select add component and select new script, and give it a name. Click create and add and then double click on the script and then a programming environment (usually mono) will open.


New Script

Name: colour_change

Language: CSharp



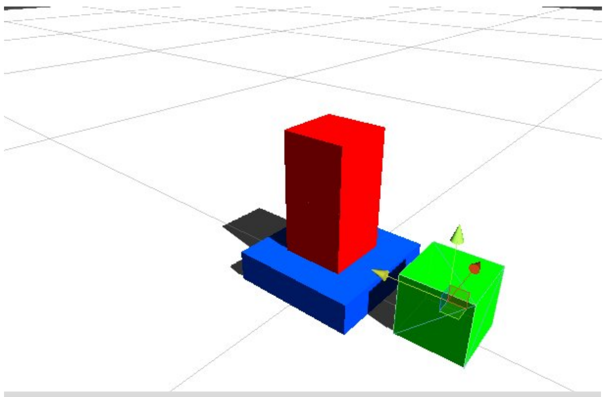
```
using UnityEngine;
using System.Collections;

public class colour_change : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

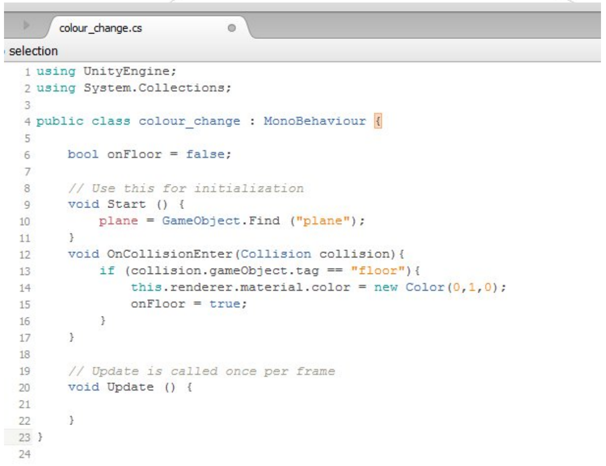
before the start function add a variable of type bool and call it onFloor, this will be what we look at to see if it is on the floor for a score later. We need to use the inbuilt OnCollisionEnter function to see what happens when the cube collides with something, use the script below and remember to save in mono. Click on the plane and then on the dropdown list labelled tag, add a tag called floor in element 0 and set the planes tag to floor.

```
void OnCollisionEnter(Collision collision){

            if (collision.gameObject.tag == "floor"){

                  this.renderer.material.color = new Color(0,1,0);

                  onFloor = true;

            }

      }
```

When it enters a collision it will see if it is hitting the floor and if it is it will set the colour to green and the onFloor variable to true. on the other cubes click on add component > scripts and the script you just made.

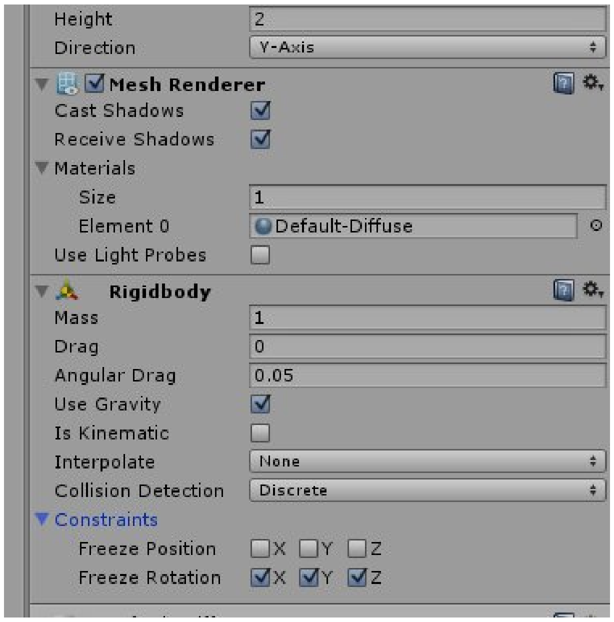
you can test it by clicking on the play button and then dragging a cube on to the floor to see if it changes colour.





```
using UnityEngine;
using System.Collections;

public class colour_change : MonoBehaviour {

    bool onFloor = false;

    // Use this for initialization
    void Start () {
        plane = GameObject.Find ("plane");
    }
    void OnCollisionEnter(Collision collision){
        if (collision.gameObject.tag == "floor"){
            this.renderer.material.color = new Color(0,1,0);
            onFloor = true;
        }
    }

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

    }
}

```

Now we need a player to control, add a capsule and give it a rigid body. and in the constraints freeze all of the rotation. this will prevent the player from falling over.

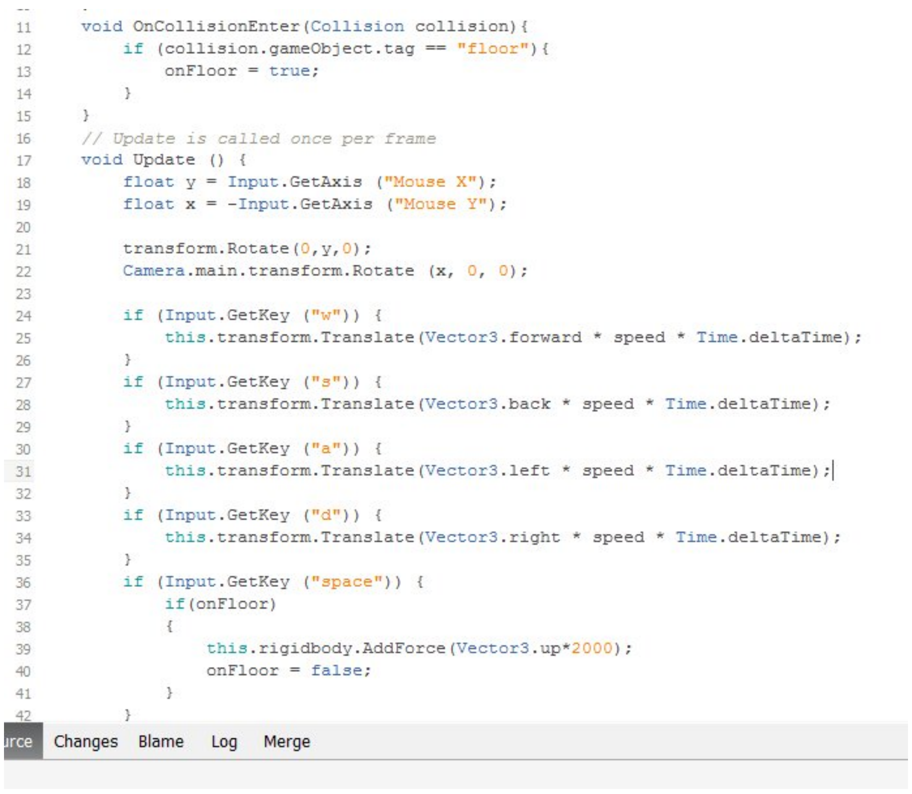


Height	2
Direction	Y-Axis
Mesh Renderer	✓
Cast Shadows	✓
Receive Shadows	✓
Materials	
Size	1
Element 0	Default-Diffuse
Use Light Probes	☐
Rigidbody	
Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	☐
Interpolate	None
Collision Detection	Discrete
Constraints	
Freeze Position	☐ X ☐ Y ☐ Z
Freeze Rotation	✓ X ✓ Y ✓ Z

create a script for the player and add the following variables

bool onFloor = false;   float speed = 3; and in the start function add Screen.lockCursor = true;

and this code



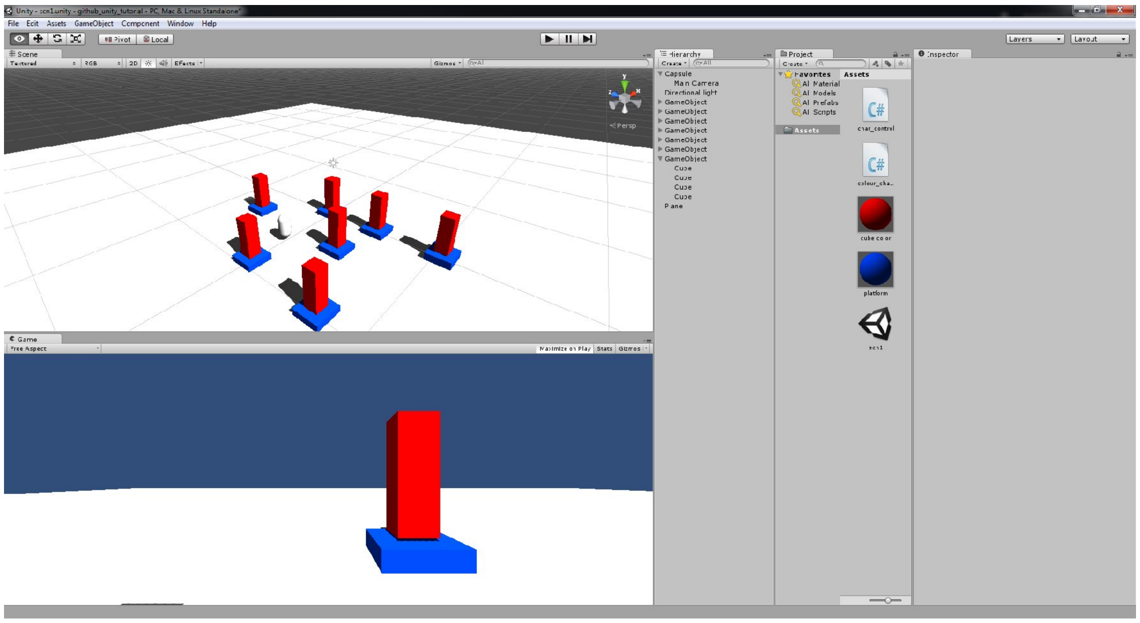
```
    void OnCollisionEnter(Collision collision){
        if (collision.gameObject.tag == "floor"){
            onFloor = true;
        }
    }
    // Update is called once per frame
    void Update () {
        float y = Input.GetAxis ("Mouse X");
        float x = -Input.GetAxis ("Mouse Y");

        transform.Rotate(0,y,0);
        Camera.main.transform.Rotate (x, 0, 0);

        if (Input.GetKey ("w")) {
            this.transform.Translate(Vector3.forward * speed * Time.deltaTime);
        }
        if (Input.GetKey ("s")) {
            this.transform.Translate(Vector3.back * speed * Time.deltaTime);
        }
        if (Input.GetKey ("a")) {
            this.transform.Translate(Vector3.left * speed * Time.deltaTime);
        }
        if (Input.GetKey ("d")) {
            this.transform.Translate(Vector3.right * speed * Time.deltaTime);
        }
        if (Input.GetKey ("space")) {
            if(onFloor)
            {
                this.rigidbody.AddForce(Vector3.up*2000);
                onFloor = false;
            }
        }
```

the if statements check for key presses and move the player. when the space key is pressed a force is applied causing the player to jump. the onFloor bool is used so the player can't jump in mid air and the collision function checks if we're on the ground. the input get axis part gets the mouse input and maps it on to the direction the player is facing and on to the cameras inclination. you will also need to parent the camera on to the player and set its position to 0,0,0 so when the player moves the camera does too.

testing the game now you can move and jump in to the blocks to push them over so they turn green. press escape to free the mouse.

copy and paste the tower game object to create multiple towers to topple.

you should have something like this



now we will add a gui text to display the score.

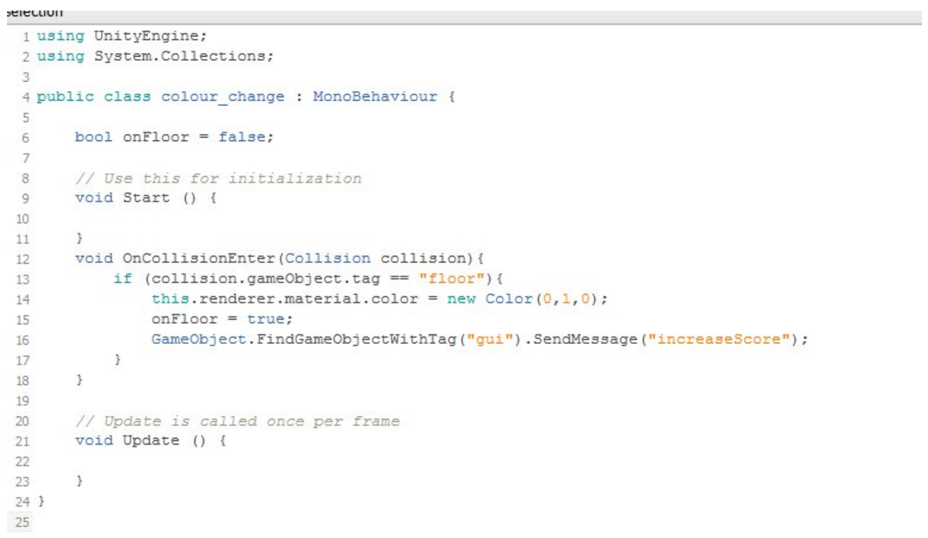
add a GUI text from the GameObject menu and set the position to 0,1,0 so it's in the top left corner.

now add a script for the guitext and enter the following code.

```
using UnityEngine;
using System.Collections;

public class score_script : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }
    float score = 0;
    // Update is called once per frame
    void Update () {
        this.guiText.text = "Score = " + score_script;
    }
    void increaseScore()
    {
        score ++;
    }
}

```

create a tag called gui and assign it to the gui text

Finally modify the colour change script to look like this



```
using UnityEngine;
using System.Collections;

public class colour_change : MonoBehaviour {

    bool onFloor = false;

    // Use this for initialization
    void Start () {

    }
    void OnCollisionEnter(Collision collision){
        if (collision.gameObject.tag == "floor"){
            this.renderer.material.color = new Color(0,1,0);
            onFloor = true;
            GameObject.FindGameObjectWithTag("gui").SendMessage("increaseScore");
        }
    }

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

    }
}
```

you can download my project from github at : https://github.com/lazerduck/Unity_Tutorial