Basic Scripting

http://unity3d.com/learn/tutorials/modules/beginner/scripting

Scripts as behaviors

http://unity3d.com/learn/tutorials/modules/beginner/scripting/scripts-as-behaviour-components

Scripts describe the behavior of a game object

Try this:

GameObjects>3D Object> Cube

Click on the cube and look in the inspector

Add Component>New Script>ChangeColor> Create and Add

Double click on the ChangeColor script in the project tab

This opens Visual Studio and shows you the script

In update type

```
gameObject.GetComponent<MeshRenderer>().materials[0].color = Color.red;
```

You can put this on any game object that has a MeshRenderer. Lets break this down:

gameObject – the game object that contains this script

MeshRenderer – the component of the game object that does the drawing

materials – the list of materials applied to this MeshRenerer that defines color and lighting properties

Look at the cube gameObject in the hierarchy and observe how it corresponds to the script.

Wait, where was game object defined?

gameObject is defined implicitly for all objects of type MonoBehaviour (this class inherits from) public class ChangeColor : MonoBehaviour

How do you do this for any type of component in this gameObject (or another gameObject)

GetComponent

http://unity3d.com/learn/tutorials/modules/beginner/scripting/getcomponent

This will return a reference to any type of component you are looking for (assuming it exists in the calling GameObject or null if it doesn't)

- `GetComponent<class>()` – looks for a component in the current gameobject of this type
- `GetComponentInChildren<class>()` – looks for the first instance of this component class in a child (e.g., as defined by the hierarchy tab)

Define some instance variable components and use them in the script

```
    private Transform myTrans;

    // Use this for initialization

    void Start () {

        myTrans = GetComponent<Transform> ();

    }

    // Update is called once per frame

    void Update () {

        if(myTrans.position.x >0)

            gameObject.GetComponent<MeshRenderer>().materials[0].color =
Color.red;

    }
```

- What if we wanted to instead change the color in the inspector during play?
    - Define an instance variable as **public**

```
public Color myColor = Color.red; // red is the default
```

- and in update change:

```
gameObject.GetComponent<MeshRenderer>().materials[0].color = myColor;
```

- now look at the inspector, hit play, move the cube in the positive x direction, and then change the color
    - (remember changes are not kept after play mode ends!)

Getting components from other game objects

The easiest ways is to

1. create a public variable
2. and then in the inspector drag and drop the gameObject that has the component that you want to reference (e.g., another cube).

Activity: Add a sphere and change its color based on the cube.

First create another GameObject>3D>Sphere

Now add another **public** variable to the ChangeColor script:

```
MeshRenderer sphere;
```

Drag and drop the sphere into the sphere slot in the inspector

Make it so that the cube is the same color as the sphere when the cube's position.x>0

Overriding Monobehaviour's functions

Every game object has a set of functions that are called in a specific way (you can of course define your own too)

Execution Order Diagram

http://docs.unity3d.com/Manual/ExecutionOrder.html

For now we are concerned mainly with how Awake, Start, Update, and FixedUpdate work.

Awake and Start

http://unity3d.com/learn/tutorials/modules/beginner/scripting/awake-and-start

1.Awake – called once

Called first, even if the script component is not enabled

Used for dealing with references between scripts and initializations

2. Start – called once

Called once the component is enabled

Won't get called again if disabled and reenabled

Update and fixed update

http://unity3d.com/learn/tutorials/modules/beginner/scripting/update-and-fixedupdate

3. Fixed update

    Used for physics objects – all physics updates occur directly after this function

    Don't need to multiply values by Time.deltaTime since fixed update is frame rate independent

    Same intervals between calls

4. Update - Called once every frame

    Used for regular updates

        Non physics objects

        Simple timers

        Receiving input

    Update interval times vary!

Timers and Framerates

If you are using something that will change incrementally each frame, multiply the value by Time.deltaTime!

Why? Consider translating an object.

```
void Update(){

      // translate by 1 each frame

      transform.position += new Vector3(1f,0,0);

}
```

Right or wrong? WRONG!

Different platforms support different framerates and will call update at different times – you want it to be based on time rather than frame rate so the object moves at the same virtual speed on every platform.

```
void Update(){

// translate by 1 each frame, given the amount of real
// time that has passed

      transform.position += new Vector3(1f*Time.deltaTime,0,0);

}
```

This generally means that you will need to tweak variables more to get the speed you want, but it will move at the same speed on all platforms and computers, regardless of what load the CPU has on it from other programs.

Instantiate

http://unity3d.com/learn/tutorials/modules/beginner/scripting/instantiate

If you want to create a procedural level like required in assignment 1, you will need to create the objects inside of a script.

First create a platform box prefab:

GameObject>3D>Cube

Scale it along z to look like a rectangular platform you could run on.

Now drag the cube down into the project tab – this creates a prefab (notice it is blue now in the hierarchy)

You can delete the cube from the hierarchy now

Now create an empty game object GameObject>Create Empty and name it Level Generator

Now create a script component on that object called GenerateLevel

```
public class GenerateLevel : MonoBehaviour {

      public GameObject basePlatform;


      // Use this for initialization

void Awake () {

      Vector3 currentPosition = Vector3.zero;

      Transform trans = gameObject.GetComponent<Transform>();

      for (int i = 0; i< 10; i++) {
            Instantiate (basePlatform, currentPosition, trans.rotation);
            //generate a random position (within a range) to be more fun
            currentPosition+= new Vector3(0,0,50f);
      }
}
```

Now add an FPSController and have fun!