Unity Tips and Tricks

During Spring 2013, GDD211, and in Spring 2014 CS319, we learned a lot about Unity. We have attempted to capture that knowledge below. Here is the template:

One-Line Description Of The Tip
by Your Name Here

Step by step instructions and/or content here.

Each tip should start on a new page. Use the Insert menu, Page Break.

Tips can be short and sweet, just a link with a quick description, or something long and involved. Extra credit will be awarded based on effort and value of the contributions. Of course, more important than extra credit is the benefit provided to future students.

Using Oculus Rift DK2 in Unity 3D
by Eddie Pantridge

1. Make sure you have the most recent version of Unity.
2. If this is the first time using the Oculus, follow the manual to complete the initial setup. (Also posted here).
3. Go to the Oculus downloads page
4. Download and install the “Oculus Runtime” for your game (here).
5. Download “Oculus Unity Tuscany Demo” for your OS (here).
6. Test the Oculus with the Tuscany Dome.
7. Download “Unity 4 Integration”
8. Open your Unity Project
9. Drag the “OculusUnityIntegration.unitypackage” into your Unity Assets folder and click the import button on the pop-up.
10. Go into the OVR folder then Prefabs folder in the Assets folder that was just created. Add the OVRPlayerController to you scene in place of your First Person Controller.
11. Build your game and test it out!

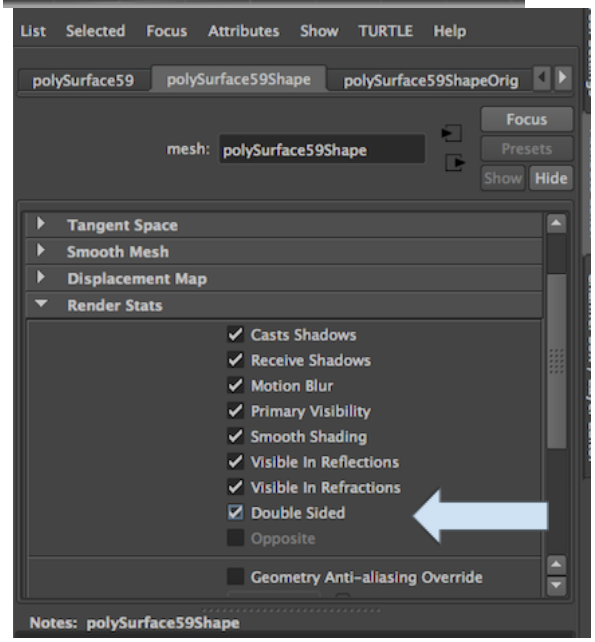
Notes:

- Oculus support was exclusively a Unity Pro feature unity the most recent version of Unity 4 Free. Make sure all members of your group are using the same, most recent version.
- When you press the play button in the Unity Editor, it will not appear in the Oculus. In order to test/play your game with the Oculus, you will need to build your game first, and run it as a standalone.

HELP Some/All of my model has holes or seems inside-out
by Connie

For folks working in maya and mostly familiar with that, you'll sometimes run into a situation where when going into unity, sculptris, or most any software package, part or all of your model looks something like this: –>

The faces are there but they aren't showing up. This is because, by default Maya, maya makes your mesh double-sided. The normals shoot both ways when rendering. Unity, Sculptris, etc don't really work that way. To fix this, you have to track down the problem faces and reverse the normals in maya. (Normals>Reverse under the poly toolset) Selecting all faces and using Normals>Conform can sometimes do the whole thing much more quickly.



To help track down the faces, you can get the mesh in maya to behave the same way as in unity by going into the attribute editor for the shape node, and under render stats, unchecking "Double Sided." If you click "reverse" here, it will render in maya as if you had reversed everything, which can make it easier to see if you have a dense mesh.

Normals sometimes get reversed when extruding, using bridge geometry, or scaling into the negative. It's sometimes better to fix this as it happens because it can be a cause of weird creasing when you smooth/use smooth preview.

This normals thing also means that non-closed shapes (like planes) are visible **from only one side**. That's not a mistake, that's just a thing. I'm pretty sure you could avoid this by using a double sided material, which you'd have to hunt down through google or program yourself.

Posting to your Hampshire Webspace
by Connie, mostly stolen from Paul Dickson (may he rest in peace)

Hampshire students have webspace at stout.hampshire.edu/~username (username being the first part of your email address. There are a couple ways to post to it, but one that's consistently worked for me is:

• Install Fugu - their not quite stable release for Lion is working for me, or, if you're on a PC, Paul recommended WinSCP. Googling alternatives for either will probably find you something that works.
• You want to connect to stout.hampshire.edu, and your username is your hampshire username. Your password is your hampshire password.
• Enter the public_html folder. It's probably bad news to mess with the other stuff, and this is the folder that will put things at stout.hampshire.edu/~you. Drag and drop what you need to here.
• Make sure your unity world was built for web, and that you move not just the HTML file, but the full contents of the folder that appears (for me this was an html file, 2 javascript files, a .unity3d file) but not the whole unity project folder, because that can be hefty.
• Paul's original documentation for this is here http://helios.hampshire.edu/~pedCS/classes/cs106Fall11/toWeb.html and includes commandline stuff if that's your jam.

†

Running your Animations in Unity - for art folks

by Connie

When exporting animation to unity, a lot of things can go wrong, but you can't even know what exciting problems you have to deal with until you can get something to be playing in a game. I'm going to assume you have something with animation imported into unity, FBX or maya file or whatever floats your boat.

- Select your animated dude in the project view. In the default unity layout, that's the big one at the bottom of your screen that says Project. If you're lost, change to the default unity layout under Window>Layouts>Default
- Check if he's animated in the preview (there should be a play button.) That should at the bottom right corner of your screen. If he is, the animation is definitely in Unity, if he's not, it's probably not a good sign, but it's best to triple check because things can end up different in the preview.
- Above the preview is the inspector. At the very top of it you should see buttons for Model, Rig and Animations. Here are the important things:
    - **Model**: The scale factor could be nuts low, and if you drag your friend into the scene and she's so tiny she's invisible, this is probably the culprit. This is an easy way to scale up, or you can use normal scaling tools
    - **Rig**: Animation Type. Your programmers may want to use Unity's new animation tools, but for you it's easiest to just set this to Legacy to use unity's old animation system. Do this if they don't care or if you're just testing stuff.
    - **Animations**: This is where you can define animations from the file you imported. Maybe the first 10 frames are a wink, then the next 15 are jumping. This is in the lower part of the interface. But that's another thing.
        - To just make sure everything's playing, make sure that you have the right start and end frames.
        - Anim Compression: If things are funky, turn this off.
        - Wrap Mode: this is in here twice - you want to change the bottom one to loop if you want looping. No idea what the first one does, you can make that also be loop if you want.
        - Add Loop Frame: Try this on and off and see which looks better. Pretty self-explanitory, but little jumps in loops are really common in unity even if you don't see that happen in Maya and it can help.
- Apply your settings and drag your object into the scene. Move it in front of a camera, add a light or two if needed, and hit play. Some motion should happen. Fiddle with the animation settings until it's right enough, or if there are big

problems, you may have messed up something in exporting.

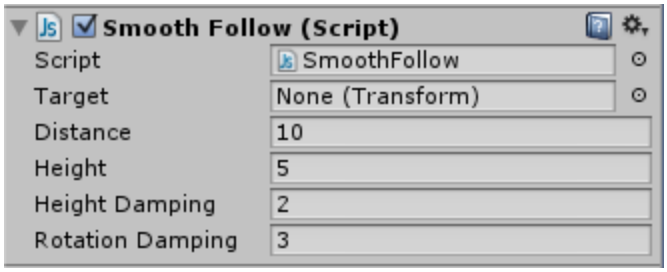
Unity (Scripts) Package

by Eddie

If you aren't big on programming, or just don't want to reinvent the wheel, you might be able to find what you are looking for in a script that already exists. I am sure that the internet has a script for every possible use, but another place to get some useful scripts is the Unity Script Package.

For this example I will show you how to use what I think is one of the most useful scripts in the package: the Smooth Follow Script for controlling the camera.

- First you need to get script package into your unity project.
    - If you are just creating the project, you can simply check the box next to Scripts.unityPackage in the "Import the following packages:" area.
    - If you already have a project just go to Assets -> Import Package -> Scripts. Make sure all boxes are checked and click Import.
- Now select the main camera in the scene.
- In the menu click Comment -> Camera-Control -> Smooth Follow
- Here is a screenshot of the box added to the inspector. This is where you tweak the script to your needs. Most scripts take from somewhere else will have a few variables in this area.



- For the Smooth Follow script, here are the variables :
    - Target: Click the little circle to the right of the text box and select the object the object that you want the camera to follow in the popup window.
    - Distance: How far away should the camera be from the object it is following.
    - Height: How far above should the camera be from the object it is following.
    - Height Damping & Rotation Damping: The higher these are, the smoother the camera motion will be. If you put these too high, the object your camera is following might move too fast and go out of frame.
- REMEMBER:
    - If you edit these variables while your game is running (you have clicked the play button) those changes will apply live, but will revert to their previous values once you stop playing.

Now your camera should follow and point at the object you chose, and you didn't have to code a thing!

Tutorial for importing objects from maya to unity with textures
by Mike Conwell

So from listening to class today and a lot of other days it has become obvious that importing from maya to unity can be an issue. Here's a tutorial that will walk anyone through how to import (geared to programmers that have a general understanding of unity).
http://www.youtube.com/watch?v=hiH5nZnkq2k

Capitalization Matters in Unity
by Zach

If you're a programmer, capitalization is very important to keep in mind when coding in Unity. In most cases, improper capitalization will produce an error and be brought to your attention right away by the compiler. However, when you wish to overwrite an inherited method, there is no way for the compiler to determine your intention. For example, if you want to overwrite the "OnCollisionEnter" method that your class has inherited so that it performs some action when it collides with another object, you would make a method called "OnCollisionEnter". If you were to name this method "OnCollision", then you haven't overwritten the "OnCollisionEnter" method; you've just created a new function that the game won't associate with collision events.

The problem with this mistake is that you haven't created any errors, which can make it very hard to find and correct if you don't know which piece of code isn't working. This is especially true when it's a simple difference in capitalization, and the misnamed method looks perfectly fine. Earlier in the semester,I tried to overwrite a method named "OnTriggerEnter" with a method named "onTriggerEnter", because that's the naming convention that I'm used to following. It took me nearly a week to figure out that the problem was my lowercase 'o' because the problem was so well-hidden. My advice to any programmers is to familiarize yourself with the conventions that Unity's code follows and not overlook capitalization when debugging your code.

How to Make 3D Letters
by Erin Candee

1. In Maya, click Create>Text>Text Curves Options.

*The Text Curves Options box is the square next to “Text” in the Create drop down menu.

2. Type whatever text you want in the Text field.
3. Choose your Font
4. For Type, you can either do Poly or Bevel. Poly turns your text into a text plane that you have to extrude. With Bevel, your text will automatically be 3D but it will have a bevel on it.

Poly:
5. If you chose Poly, select Quads and then click Create.
6. Drag select all of the letters.
7. Right mouse button drag to select Face.
8. Select all of the faces
9. Click Edit Mesh> Extrude
10. Left mouse drag the Z (blue) arrow to make the letters 3D.

Bevel:
5. If you chose Bevel, click Create.
6. If you don’t like the way it looks, you can go back and change the outer and inner bevel styles.

*i tried adding screenshots but it wasn’t working. i will try later on another computer.

How to Put a Unity Game on Kongregate
by Sam Alexander

1. Go to www.kongregate.com
2. Create an account at top of screen
    a. Either click Register or sign in with facebook
3. After your ID is setup login
4. On the homepage hover over games and then navigate down to UPLOAD A GAME under Developers
5. Fill out the details of the game; click continue
6. Under Game File Browse to your **Web Player** file and choose that
7. **You must upload a Game Icon or it will not upload**
    a. The picture file has to be small
8. Check off all 4 checks under the licensing agreement
9. Click upload at the bottom
10. When it brings you to the next page, click publish
11. Good job, you published your own game

How to add 2D billboards to a Unity World
by Ryan and/or Chris

Describe how you added the 2D image in Ryan's Team's Beauty world (the hint to right click) or the 2D images in Hero of Charring Cross. Include code samples.

~~~~~~

The 2D images that you see appear on the screen are actually part of Unity's built in OnGUI() function. This function works much like Unity's other built in functions (i.e. function Update()) in that it has build in characteristics. the various methods that you can use within this function are easily located on the unity scripting reference here. However the one we will use for the purpose of the sample code is the DrawTexture function.

Example (Javascript):

```
	var TextureExample: Texture; /*declares a variable that you can assign a texture
to later*/

	function OnGUI() {	//start of the GUI part of the code
GUI.DrawTexture(Rect(Screen.width/2,Screen.height/2,100,100),TextureExample);
/*will draw the texture TextureExample with a rectangle behind it in the middle of the
screen at 100% of the scale of the texture.*/
	}//end gui
```

END EXAMPLE

All this code does is diplay whatever you assign Texture Example to be in the middle of the screen with a box around it, hope this helped with any gui needs. Feel free to ask me if you have more questions

-Ryan

~~~~~~

How to Refer to another Object's Script in C#
by Zach and Adam Roy

Unity scripts are all children of the MonoBehaviour class, which in turn is a child of the Component class. As far as I'm aware, everything you attach to an object is, somewhere further up its family tree, a Component. When you have an event that allows you to grab an object's collider, such as a collision or trigger event, you have access to that entire object. The trick is knowing how to find each part of that object. Some, such as its transform, are easy to find: other.transform is all it takes. Others, such as scripts that not every object will have, don't have an easy to access variable reserved just for them. You have to use one of the methods inherited from the Component class to find these other Components.

The method that I just learned to use is called "GetComponent", and once you understand how to use it this method can be very helpful. First, it takes a type as its argument. I was able to have it search for my specific script by typing "GetComponent<SCRIPTNAME>()". This will have whichever Component calls this find a SCRIPTNAME script, provided one is attached. If it can't find any SCRIPTNAME scripts, it will return NULL, so unless you can guarantee that the object calling this method has the desired script attached, check the return value before doing anything with it.

In summary, to find and store a MonoBehaviour named "myScript" attached to a Component stored in the variable "other":

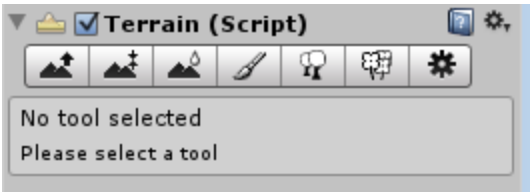
```
MyScript obj = other.GetComponent<MyScript>();
```

C#: myScript x = (myScript) other.GetComponent<myScript>();
Javascript: var x: myScript = other.GetComponent(myScript);

Using the Unity Terrain Tool
by Ryan

The terrain generator is one of the most awesome tools i've encountered so far in unity; and its super easy to use! to start simply go to "Terrain" ---> "Create Terrain" at the top of the screen in the editor. From there you will receive a GIANT white slab of blank "Terrain".

After clicking on the blank terrain You'll encounter a side window containing your usual values (Size, etc) and you should also see a window like this:



The various icons correlate to the different tools you can use. The first tool is the terrain tool, it allows you to make hills and valleys by choosing an opacity, and a brush size. You can then “draw” the hills into the terrain by holding down the left mouse button.

The second and third tools exist to help you smooth out the height of the hills so they dont turn into jagged peaks. I don't have much experience with these tools.

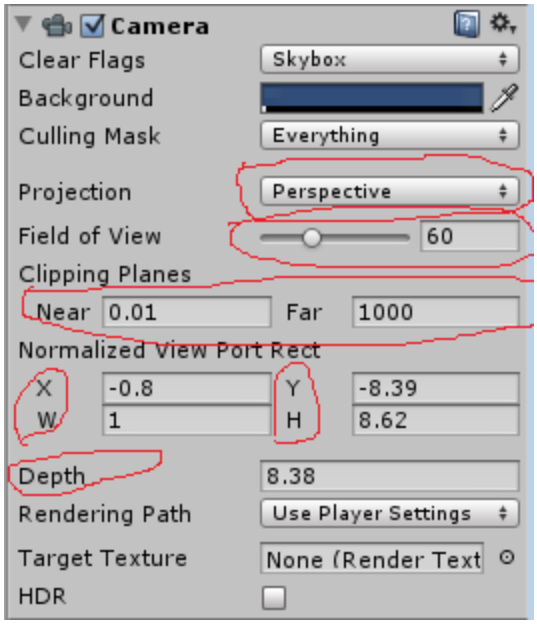
The fourth tool allows you to paint cool textures onto the terrain (I.E. moving grass).

The fifth tool is designed for placing trees/bushes/flowers/rocks etc.

And the last one is the settings on the actual terrain generator.

Mini Maps (At last)
by Ryan

The daunting task of the mini map has seemed, until now, a giant task to approach. But let me reassure you, it's not nearly as daunting as it seems. Besides your main camera, in your scene make a separate camera. For good naming practices you should name it Minimap, but honestly you can name it whatever you like. Go into the “Game” view as opposed to the “Scene” view as this will be far easier to see what the camera is looking at. You should see, on the camera something that looks like this:



I have circled (crudely) in paint, which values you should fiddle with in order to obtain the mini map and i will explain what each value does. (These are my values by the way for my mini map).

Projection: You have two options here, Perspective and Orthographic. Perspective will show you everything in between the camera and yourself, Orthographic will show you only general land features, excluding some details. Fiddle with whichever one you think fits your game more.

Field of View: This will determine the... well... field of view of the camera, adjust this value to your liking.

Clipping Planes: I am not entirely sure about this one, but i am fairly certain that it has to do with the rendering distances away from the camera. When i lower it, it no longer renders the roofs of buildings in my game and only shows me the floors and objects in them.

Normalized View Port Rect: THIS IS VERY IMPORTANT. These values essentially box the view of the camera off separately from the main camera, I am not entirely sure how the values correspond but if you fiddle with them you will see the box get bigger and smaller and appear on different corners of the screen. (Once again, “Game” view is fantastic for this).

Finally, Depth: This one is fairly self explanatory and displays the depth of the camera view.

With all the other values, you're on your own, I find they either did nothing useful or were all too confusing.

Heres what it looks like in my game:

Instructions for uploading unity project to student webspace at Amherst College:
By Aashish Karki

You may not have realized it, but every Amherst College student has a webspace. The address of the webspace goes like this:

www.amherst.edu/~yourAmherstCollegeUserName

Below is instructions to access the webspace on a mac. This only works if you are on Amherst College campus.

- Go to finder. Click “go” and then “Connect to Server”.
- Type : smb://unix-mac.amherst.edu
- It should prompt you with two folder to connect to. Choose the one labelled WWW.
- Tada! This is your space. By default, the address of your website points to index.html. You can add folders/stylesheets/scripts in this folder and modify index.html so that it uses them.

Important details for uploading to Hampshire College’s Home Spaces:
By Matt K

- To log in, use a campus computer to connect to http://stout.hampshire.edu/, use your HampID to log in, and then place anything you need in your home folder.
- Your website address will be http://home.hampshire.edu/~<HampNet ID>.

JS Library to prettily display your code on your website [Basic HTML and CSS stuff]
By Aashish Karki

If you want to display code in browser and want it colored and tabbed, consider using google’s code prettifier. It is basically a .js library and all you need to make your code prettier is import the library in the html file header and add *class="prettyprint"* to whatever or or <p> or wherever your code is. It makes the code *so* much readable. The library does not seem to add linebreaks though. So you will have to do that yourself using
.

Here is the link to the .js library you can import from:
https://google-code-prettify.googlecode.com/svn/loader/run_prettify.js

More detailed instructions on the library:
http://google-code-prettify.googlecode.com/svn/trunk/README.html

Singletons in C#

by Adam Roy

Singletons are helpful in Unity to store resources such as AudioClips. They can be made like this.

```
public class AudioManager : MonoBehaviour
{
    // Singleton pattern
    private static AudioManager _instance;
    public static AudioManager instance { get { return _instance; } }

    public AudioClip[] clips;

     void Awake ()
     {
         if (_instance != null) Destroy(_instance.gameObject);
         _instance = this;
     }
}
```

Stick this on the Main Camera and fill `clips` with your sounds, then in another script you can use:

```
void Foo()
{
     AudioManager.instance.PlayClip("myClip");
}
```

where `PlayClip(string)` searches through `clips` by name and plays the corresponding if present.

Events in C#

by Adam Roy

Events are super useful for defining relations without coupling objects together.

Syntax:
Events are variables that are defined in terms of a `delegate`.
A `delegate` is nothing more than a predefined method signature.

To make them easy, use System.Action for void methods with no parameters, and System.Action<T1 [, T2, …]> for those with one or more parameters and no return type.

Rather than explain further, I'll show an example.

Example:

```
public class Button : MonoBehaviour
{
	public int ID;
	public event Action<int> buttonClicked;

	void OnMouseUpAsButton()
	{
		if(buttonClicked != null) buttonClicked(ID);
	}
}
```

Note you must do a null check first to see if it has any listeners. Client code can do this:

```
public Menu : MonoBehaviour
{
	public Button btn;

	void Start()
	{
		btn.buttonClicked += OnButtonClicked;
	}

	void OnDestroy()
	{
		btn.buttonClicked -= OnButtonClicked;
	}
```

```
	void OnButtonClicked(int id)
	{
		if(id == 0) // Do something with button 0
	}
}
```

You register methods that match the event's delegate using `+=` and unregister with `-=`.

Use events to prevent unnecessary references. Doing the example without events would mean each `Button` would need a reference to a `Menu`.

How to Structure Game Code

by Adam Roy

This is my general practice and it works well for small games. Start with a script called `GameManager` attached to the Main Camera. `GameManager's` responsibility is to perform interactions between the parts of the game.

From there, define each subsystem, and represent each in `GameManager` with a public variable or array. Each subsystem can have it's own inner workings that should not matter to `GameManager.`

The GameManager can do something each time Update() loop to ensure the game is progressing as intended, and pass information between each of the subsystems you created.