CAMOSUN COLLEGE

COMPUTER SCIENCE DEPARTMENT

ICS 123 - GAMING AND GRAPHICS CONCEPTS

LAB 1 – INTRO TO UNITY

DUE: DEMO BY END OF LAB
You must work individually for this lab.
Submission Instructions are on the last page of the lab.

Objective:

To introduce the students to the Unity game engine, familiarize them with the Visual Editor and MonoDevelop, and introduce the students to Unity's Script Components.

Preparation:

Supplemental Video Tutorials:

https://unity3d.com/learn/tutorials/topics/interface-essentials

Background:

1. What is Unity?
   Unity is a game engine used to create video games targeting a variety of platforms. It provides a set of game development tools that eases the process of developing games. It contains tools that your game can use right away (ex. physics simulation, shadows) and also allows you to add your own custom environment and behaviors to your game.

2. What are Unity's advantages?
   One advantage is the extremely productive visual workflow. The visual editor is used to lay out the scenes in your game and to tie together art assets and code into interactive objects. Another advantage to Unity is its high degree of cross-platform support. You can deploy your game to Windows, Mac OS, Linux, the Web (WebGL), Mobile, and Consoles. In addition, Unity's development tools run on Windows, Mac OS, and Linux.

3. What are Unity's disadvantages?
   Like any platform, Unity isn't without its faults. Unity's component system, used to create game objects, can become difficult to manage, especially in complex scenes. Unity also doesn't support linking to external code libraries. Libraries used in projects must be manually copied in. Finally, Unity prefabs (will be covered later in the course) are awkward to edit.

4. Is Unity used in the real-world?
   The answer is, yes! ***Gone Home***, a popular storytelling adventure game, was developed with Unity. The ***Temple Run Trilogy***, a hugely successful mobile endless runner franchise, was developed with Unity. ***Assassin's Creed: Identity***, a mobile action adventure game based on the Assassin's Creed franchise also was developed with Unity. This is a real tool

that real companies use!!! See http://madewith.unity.com/games for more examples.

Installation (for home/laptop):

1. Go to https://unity3d.com/ and click on the ***Get Unity Now*** button.

2. On the next page, you want to download the Personal Edition (Free Download). Click on the ***Download now*** button. Finally, on the next page, click on the Download Installer button and save the installation file (at the time of this lab, Unity is at version **5.5.0**).

3. Double left-click on the installation file to begin the installation process.

4. Once the installer has started, accept Unity's Terms of Service and select the 64 bit architecture. Hit Next. You will then get a *Choose Components window* like Figure 1 below:

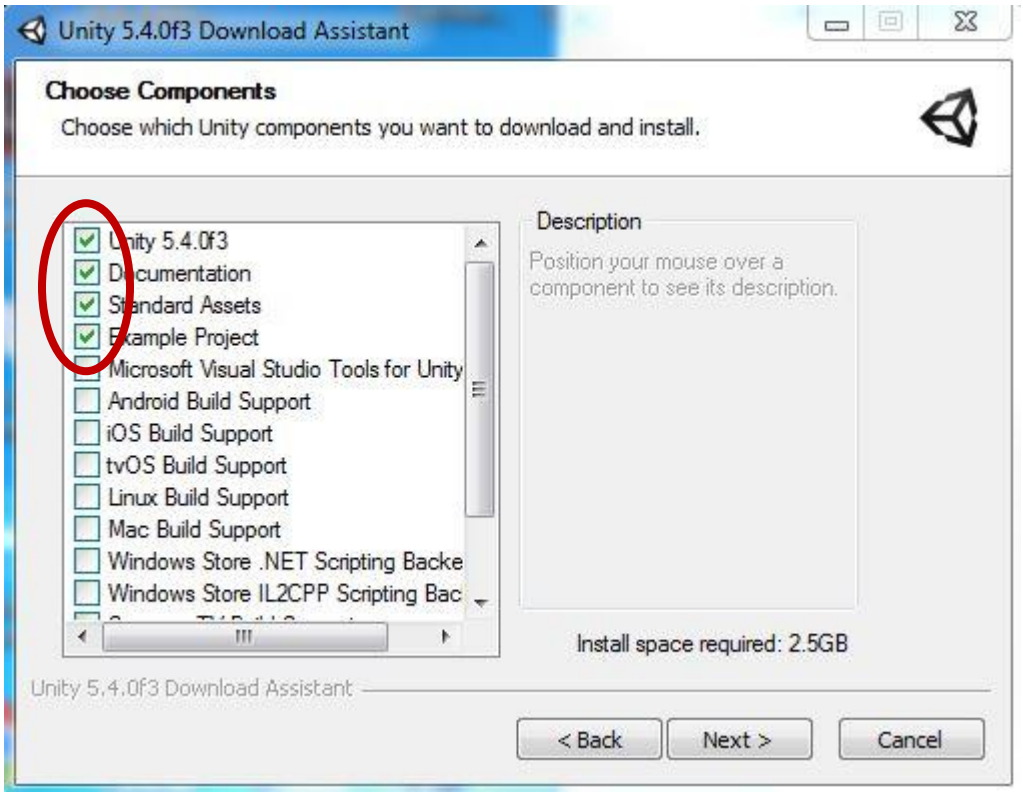

Figure 1: Choose Components Window

Check those items as shown in Figure 1. You will probably need to **uncheck** *Microsoft Visual Studio Tools for Unity* and **check** *Example Project*. If you are installing Unity on a platform other than Windows (or plan to deploy to a platform other than Windows), make sure their Build Support is checked (Windows support is built-in). For this course, demos will be done on the Windows platform (or if you wish to demo on for example, your own Mac Laptop, then ensure *Mac Build Support* is checked). Notice that the installation will take approximately *2 – 3 GB of space*. Ensure you have the free space available! Hit *Next* when done.

5. The rest of the installation should be straight forward. If you have a firewall running on your machine, it may ask you to allow access. Ensure you do!

6. Finally, when the installation has finished, launch Unity.

Launching Unity for the First Time:

1. When you launch Unity for the first time, it will ask you to sign in with your Unity account. To use Unity, you have to download a license via your account (don't worry, it's free!). Click on the '*create one*' link to create an account (see Figure 2 below):

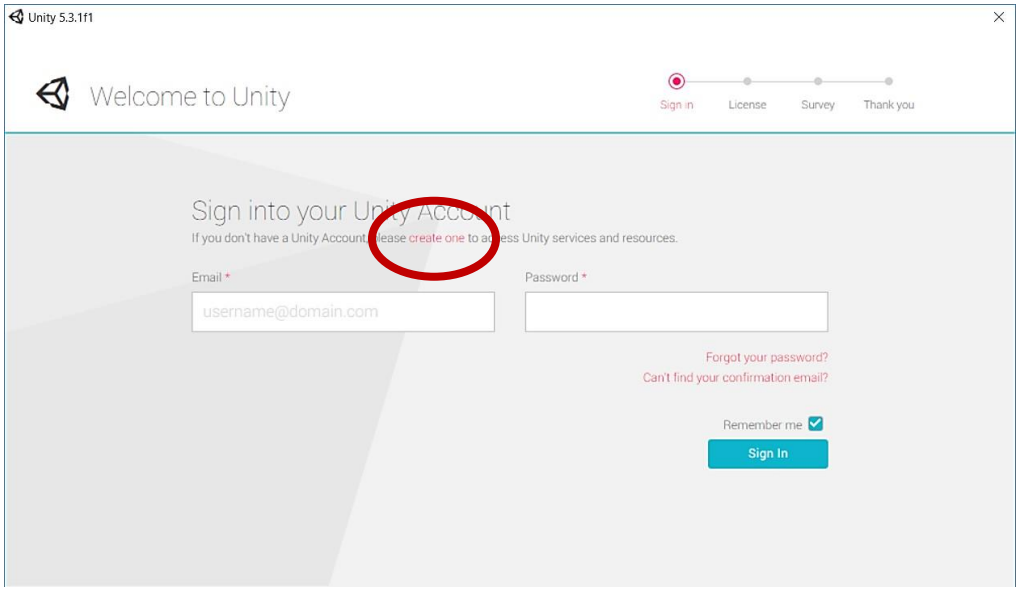

Figure 2: Welcome to Unity Window with 'create one' link circled.

2. This will open a webpage with a form you need to fill out. You will then be sent an email to confirm your account.

3. Log into Unity. You will be presented with a License Management Window (Figure 3). Make sure to select ***Unity Personal Edition***.

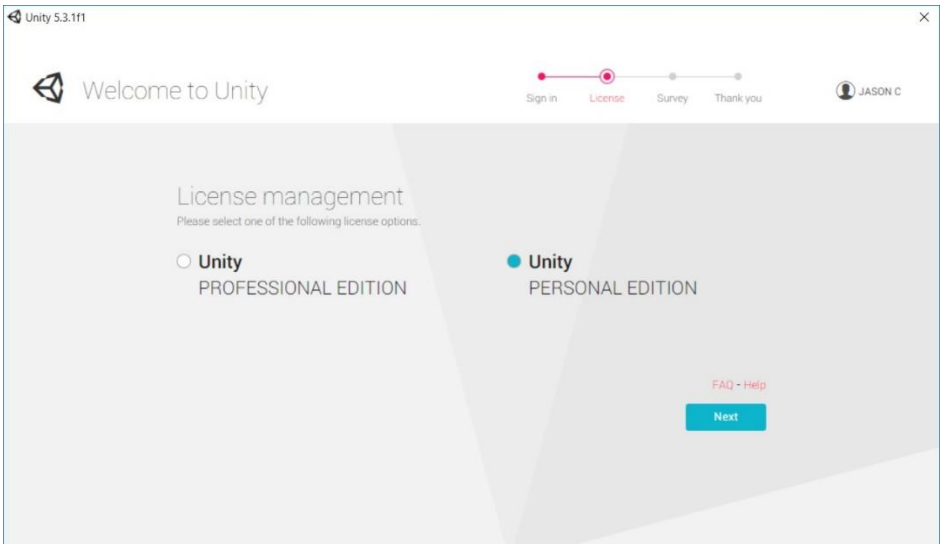

Figure 3: License Management Window

4. Hit Next. The next window asks you to fill out a short survey. Fill out the survey and hit

the OK button at the bottom. Finally, in the next window, hit the *Start Using Unity* button to activate your license.

Tasks:

Follow the tasks carefully. There is a lot of foundational material to absorb in this lab!

Tasks Part 1 – Getting to Know Unity's Visual Editor:

1. Once you have the license installed and Unity launched, you will see a Welcome window like that of Figure 4 below:

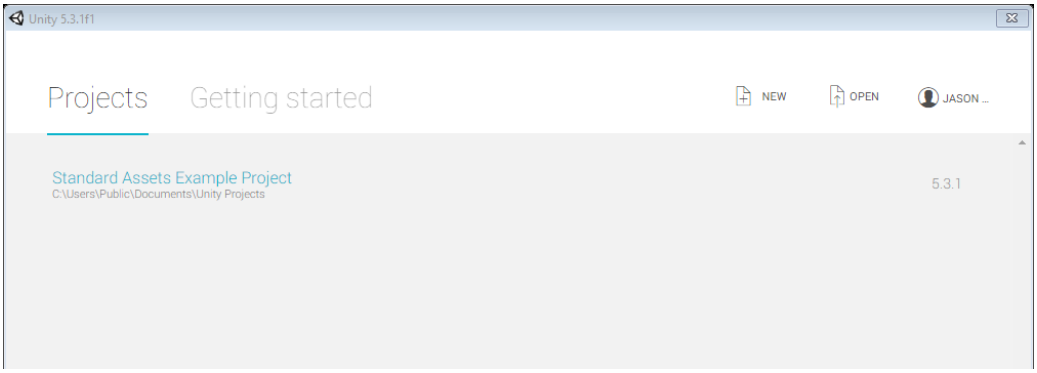

Figure 4: Unity Start Screen

Click on the ***Standard Assets Example Project***. This will cause Unity to load the Assets for the Project and launches the Visual Editor (Figure 5 below):

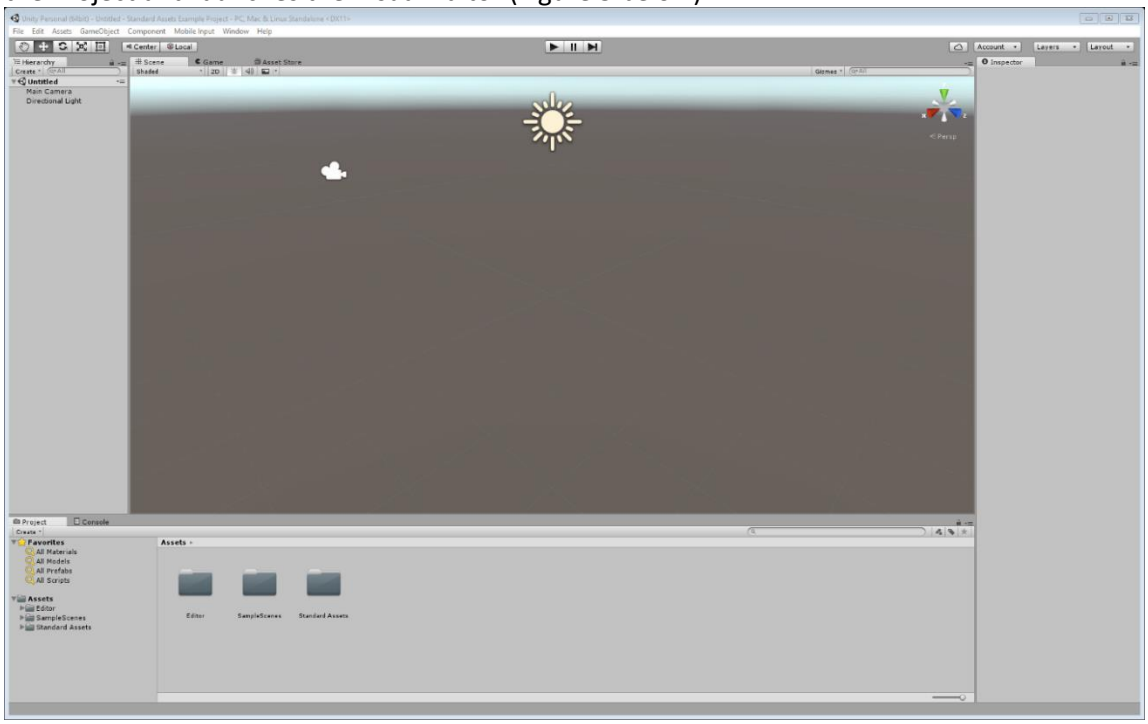

Figure 5: Unity's Visual Editor

Let's quickly look over each section of the interface:

- Across the top is the Toolbar (below the Menu bar)
- On the left is the Hierarchy Tab
- In the center are the Scene, Game, and Asset Store Tabs (you can alternate between each by clicking their title). By default, the Scene Tab is activated (like in Figure 5).
- On the right is the Inspector Tab
- On the bottom is the Project and Console Tabs (you can alternate between each by clicking their title). By default, the Project Tab is activated (like in Figure 5).

2. Let's get an idea as to what each of these sections do. For starters, the **Project Tab** allows you to browse through all the files in the Project. Move your mouse down to this tab now and expand the *SampleScenes* folder (click the little arrow on the left) under *Assets* and click on the *Scenes* folder (Figure 6 below):

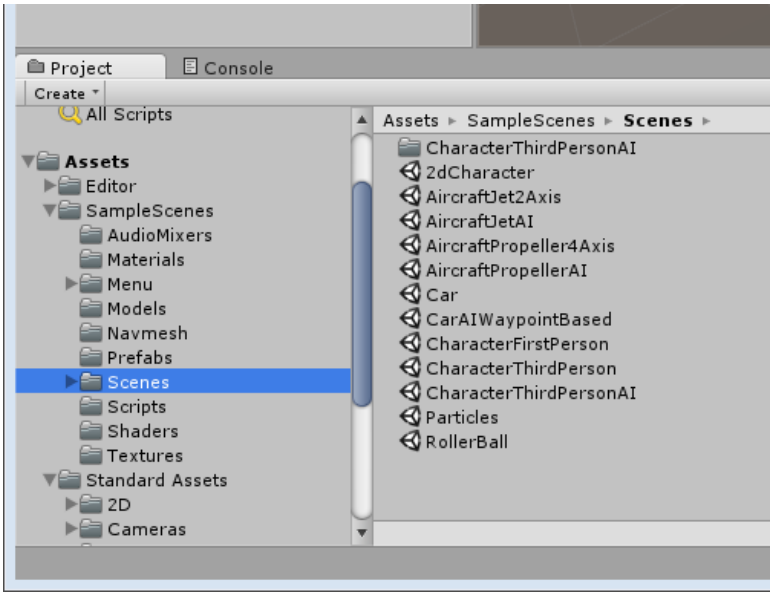

Figure 6: Project Tab

3. In the lower right corner of the Project Tab is a small scrollbar. Moving the circle in that bar changes how the icons appear in the Project Tab. Move the circle all the way to the left, and the icons are very small and appear as a list (Figure 6 depicts that). Move the circle all the way to the right, and the icons appear very large (Figure 7):

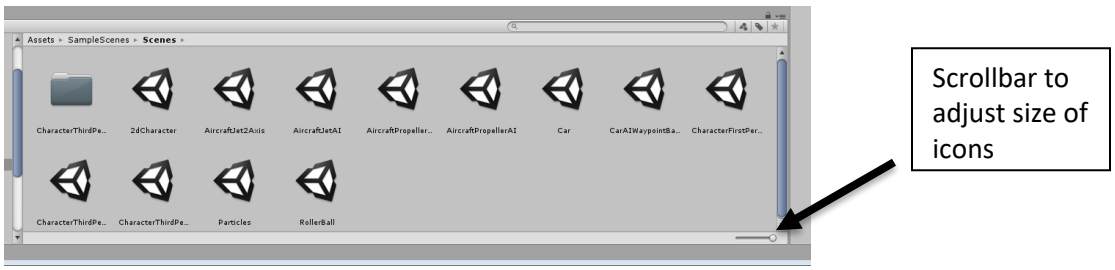

Figure 7: Icon Size Adjust Scrollbar in the Project Tab

4. In this folder, you will see a sample scene called *Car*. Double-click on that to load the scene. You should see a 3D model of a car in the Scene Tab! Your Visual Editor should now appear like that in Figure 8 below:

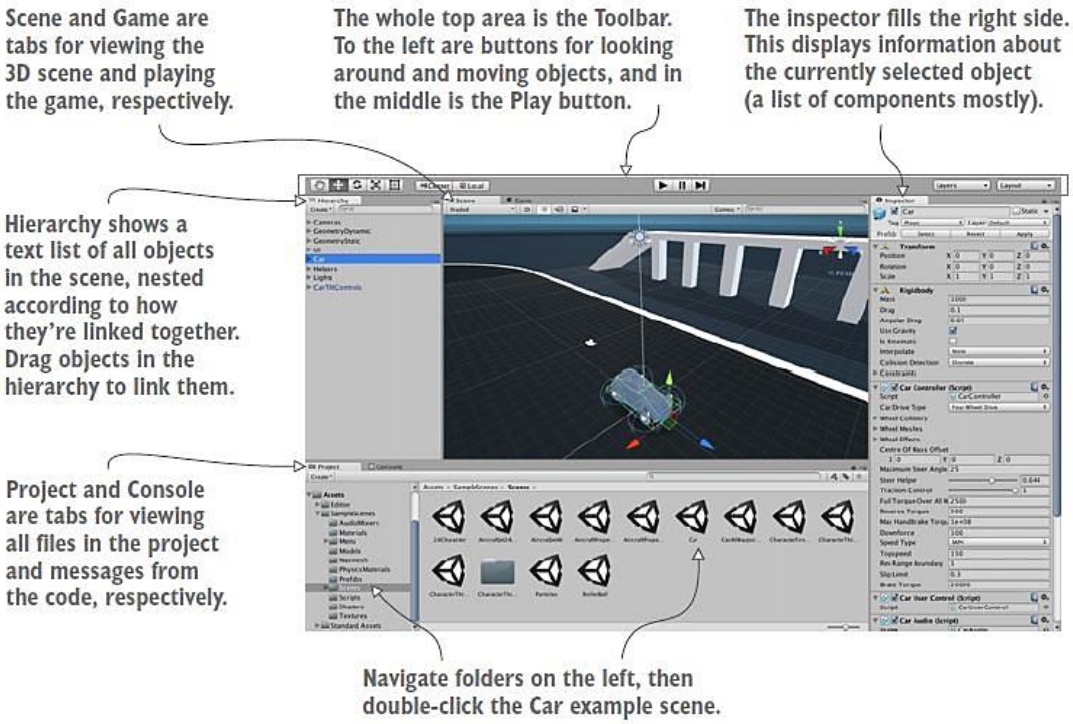

Figure 8: Visual Editor after loading the Car Scene
(Credit: Figure 1.9 from *Unity in Action*, used with permission)

5. The most prominent thing you will notice is in the center of the Visual Editor is the **Scene Tab** and in the centre of this, a **mesh object** (the car). Objects (like the car) in a 3D space are made out of connected lines and shapes, i.e. a "mesh". You can rotate the Scene view by holding down the right mouse button and moving the mouse around (you will see other mesh objects). You can zoom in and out by rotating the mouse wheel. If you zoom out far enough, you will also notice a light source and camera (see Figure 9 below):

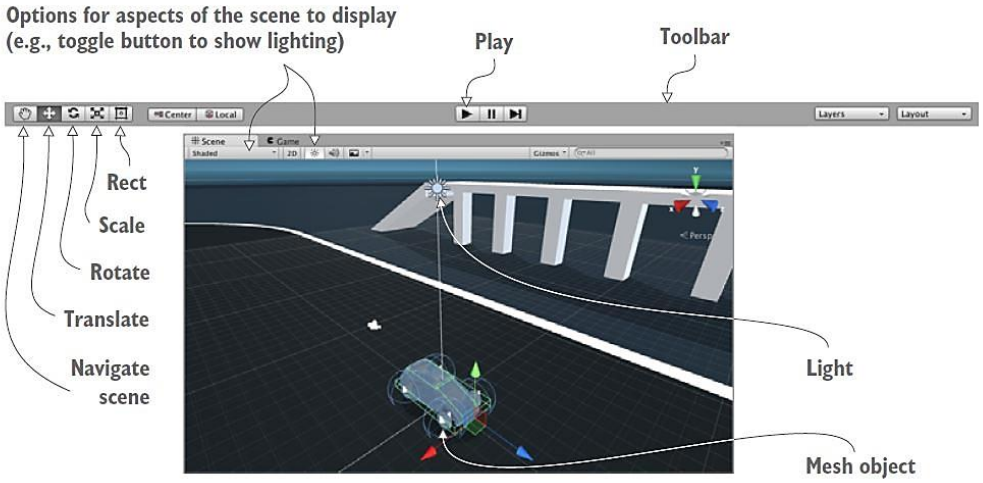

Figure 9: Scene View and Toolbar
(Credit: Figure 1.10 from *Unity in Action*, used with permission)

6. The five buttons on the top left of the toolbar are for scene navigation and transforming objects. Click on the farthest left button (the hand). This is the *scene navigation tool*. It allows you to do three things:
    a. Holding down the left mouse button allows you to *move* (or *translate*) the camera around the scene.
    b. Holding down the Alt key and the left mouse button allows you to *orbit* (or *rotate*) the camera around the scene (or you can hold down the right mouse button like in step 5 above).
    c. Holding down the Alt key and the right mouse button allows you to *zoom* (or *scale*) the camera in the scene (or you can use the mouse wheel if you have one).

    Try these now!!!

7. The next three buttons are for transforming objects. Navigate around the scene until you find a stack of blocks. *Hit the button to the right of the hand* (this is the ***move (or translate) button*** for objects). Now left click on the **top** block. You've now selected the object and should see the x, y, and z coordinate arrows overlaying the object in the scene as in Figure 10:

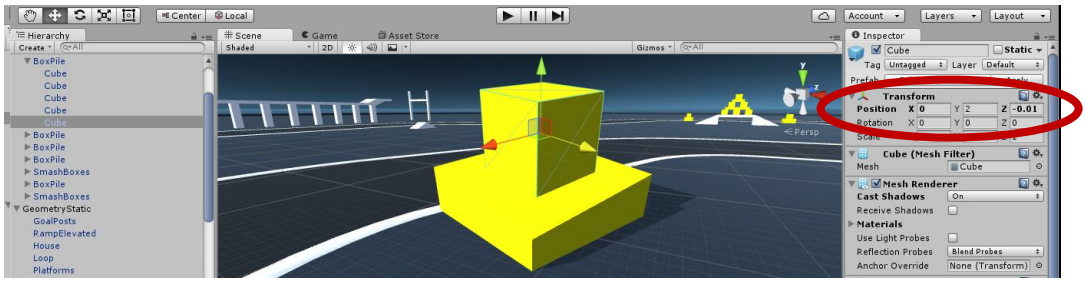

Figure 10: Cube Selected with the Move Transform Tool

8. Notice that the object you selected is highlighted in the **Hierarchy Tab** on the left. In the Scene Tab, you can now move (or translate) the object along one of the x, y, or z planes by dragging the appropriate arrow. For example, clicking on the red arrow moves the object along the x coordinate plane. Try moving the object now.

9. You will notice that the **Inspector Tab** contains information about the currently selected object. In fact, when you move the object along the x-axis, the Transform window updates the x position (circled in Figure 10). Try moving the object in the y and z directions and see how the Inspector Tab updates the coordinates in the transform window of the Inspector Tab.

10. Now click on the next button to the right of the move button in the toolbar. This is the ***rotate (or orbit) button***. Instead of the three arrows, a bunch of circles surround the object (Figure 11). In the Inspector Tab, the Rotation value of the object will change. Try rotating the object now.

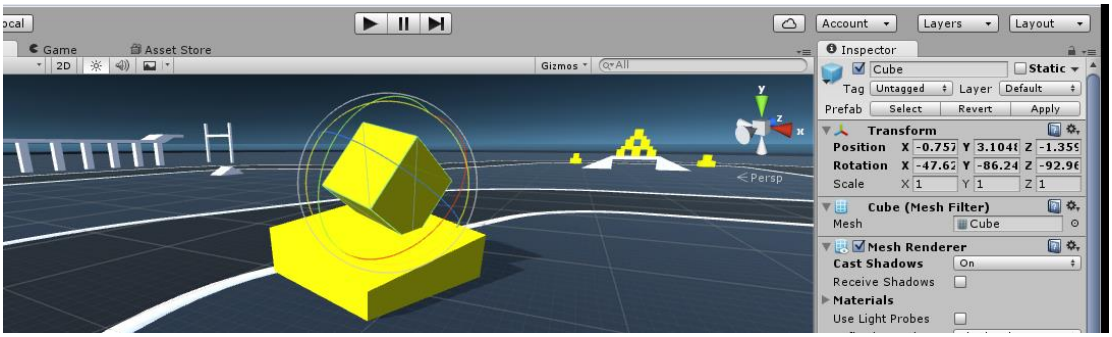

Figure 11: The Rotate Transform Tool Selected on the Object

11. Click on the next button to the right of the rotate button. This is the ***zoom (or scale) button***. You should see something similar to when the move button is activated except instead of arrows, there are 3 small squares. Clicking and dragging any one of those squares increases the scale of the corresponding x, y, or z dimension of the object. This will also change the Scale value in the Transform window of the Inspector Tab. Dragging on the grey box in the center of the zoom tool in the Scene Tab scales the entire object in all three dimensions (Figure 12).

    Try changing the scale of the object now!

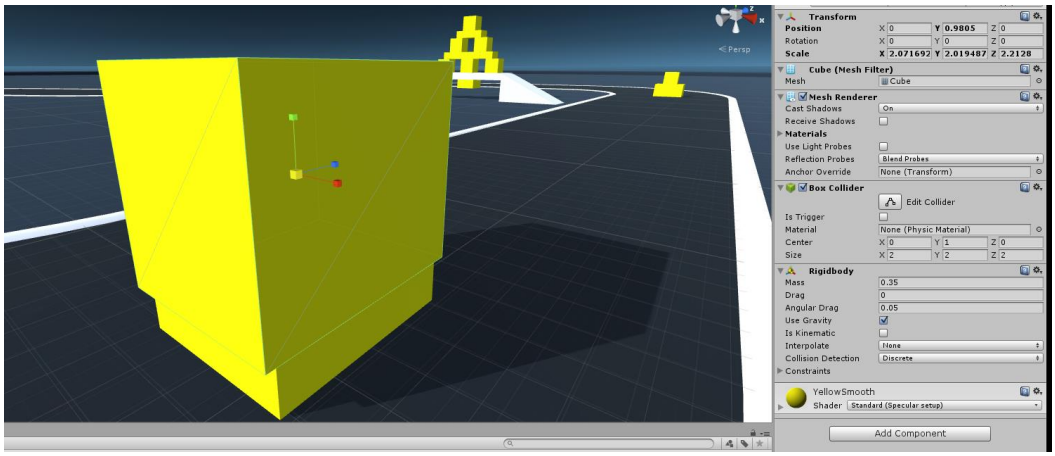

Figure 12: The Zoom Transform Tool

12. The last button in the Toolbar is the *Rect tool*. It combines movement, rotation, and scaling for 2D graphics. Ignore this for now; we'll learn about it later in the course.

13. As shown in Step 8, the **Hierarchy Tab** lists all the objects that make up the Scene. You may have noticed that the objects are listed in a hierarchical folder like structure. For example, the Cube you have been Transforming is a child object of a BoxPile. In other words, the Hierarchy Tab shows the hierarchical object linkages. If you left-click on the BoxPile parent folder, you will notice in the Scene Tab that all the cubes that make up that BoxPile are now selected (Figure 13) and the Inspector Tab shows the entire BoxPile as the selected objects. Now hit one of the transform buttons and transform the objects. You will notice that all the Cubes that make up the BoxPile transform together. The objects are all linked together!!

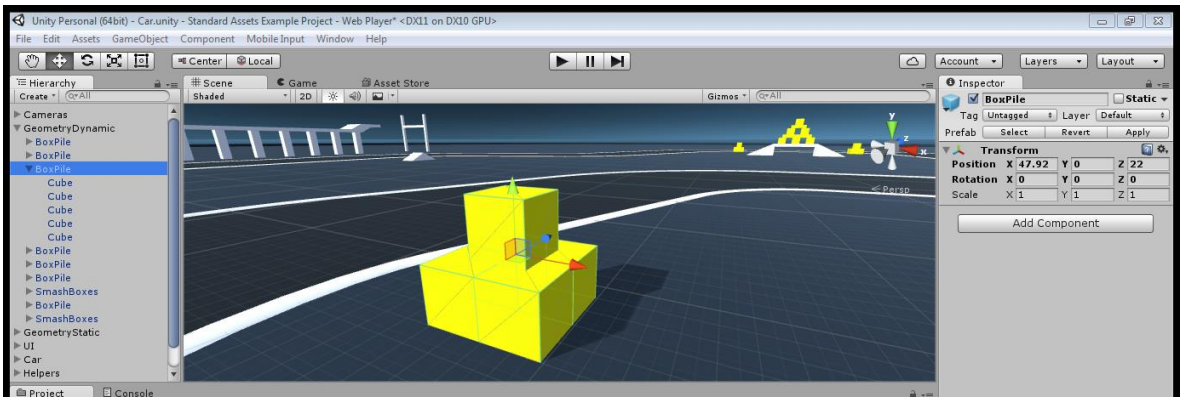

Figure 13: The Entire BoxPile Selected

14. As explained in Step 9, the **Inspector Tab** shows information about the currently selected object (or group of linked objects). Each of the windows in this tab correspond to a component that makes up the object. All game objects will have the Transform component. Some objects will have several components, including a script component. Script Components will be discussed in *Tasks Part 2* of this lab.

15. As explained in Step 2, the **Project Tab** contains a list of all files that make up the project. Like the Hierarchy Tab, it is listed in a hierarchical structure. Unlike the Hierarchy Tab, these are the list of Files and Folders in the Project, not objects, and correspond to the files and folders on disk. If you right-click on the parent Assets folder and select *Show in Explorer*, you will see where all the files and folders are stored on disk in Windows Explorer. **You should however, only delete or add files within Unity (not using Explorer)**. That way, Unity will stay in sync with the files and folders actually on disk.

16. Finally, the **Console Tab** is where your debugging messages go OR where Unity will emit its own error messages. We'll be using this Tab as we code with Unity in the course.

17. So, now the fun part!!!!! How do you play the game? Easy, simply press the Play button at the top centre toolbar above the Scene Tab. This will automatically switch the view to the Game Tab. Here, you can use the arrow keys to accelerate, decelerate and turn the car. Try interacting with other objects in the scene and see what happens!! When you are done, press the play button again or close Unity (don't save any changes).

Tasks Part 2 – Unity Programming

Background:

Unity's Programming Languages:

Whereas Unity's Visual Editor in Part 1 is used to place and manipulate objects in a game, program code is used to define their behaviours and interactions. Natively, Unity is designed to be coded with either C# (C-Sharp) or UnityScript. C# was developed by Microsoft and has syntax very similar to Java. UnityScript is based on JavaScript. There are advantages and disadvantages to using each. That being said, most of the documentation, examples, and tutorials on programming in Unity are in C#. Therefore, in this course, we'll use C#.

Unity's Script Components:

Game objects are made up of a collection of components. One of these components can be a Script component. A script component inherits the `MonoBehaviour` base class – a class that defines how the script component attaches to a game object. There are two methods you can override: `Start()` which is called once when the object is loaded in the scene and `Update()` which is called once every *frame* (each cycle of the main game loop is called a frame).

1. Create a new project in Unity. Go up to the File menu item and select New Project (**NOT** new scene) if already in Unity or select the New button from the Welcome window.

2. A window will pop up asking you to name the project and where you should store it. Name the Project 'ICS 123 Lab 1 Part 2' and store it on a thumb drive. Leave the Project as 3D and do not load any extra asset packages. Keep Unity Analytics off.

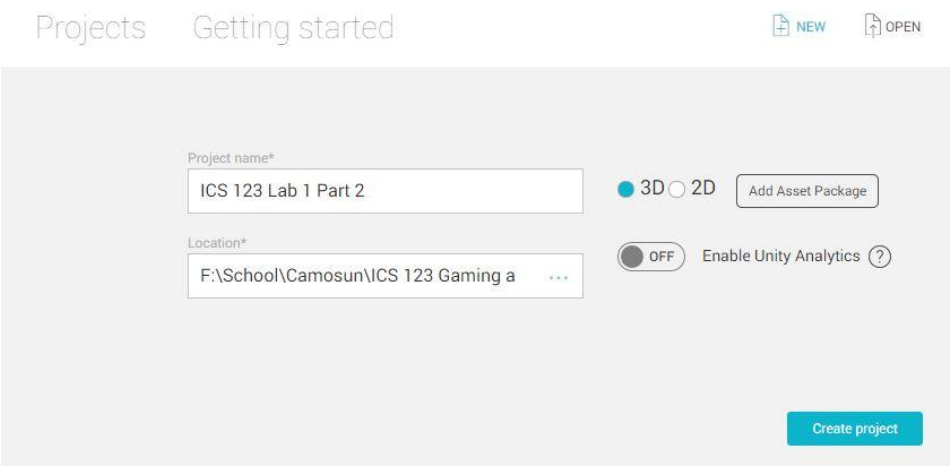

Figure 14: Creating a New Unity Project

3. Hit the **Create project** button. Unity will spend several seconds creating a new empty project.

4. Once the empty project has finished loading, go up to the *Assets* menu item and select Create -> C# Script. Name the Script 'ICS123Part2' (without the single quotes and without spaces). Your project tab should look like Figure 15 below:

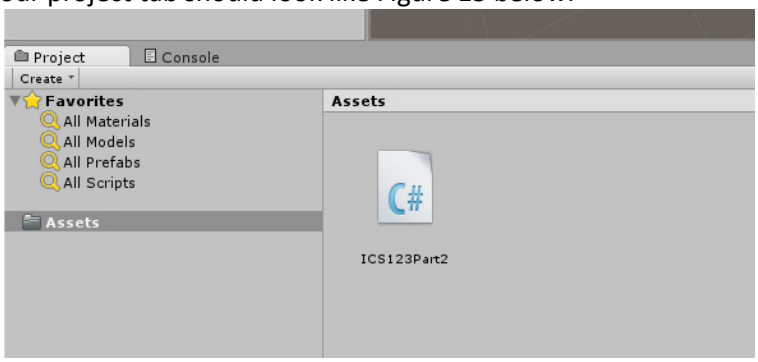

Figure 15: A new C# Script Component Created in the Project

5. Now, double left-click on the Script. This will open the Script in **MonoDevelop**, an open-source, cross-platform IDE for C#. MonoDevelop is bundled with Unity. You create empty

script files in Unity's Visual Editor but you edit them in MonoDevelop.

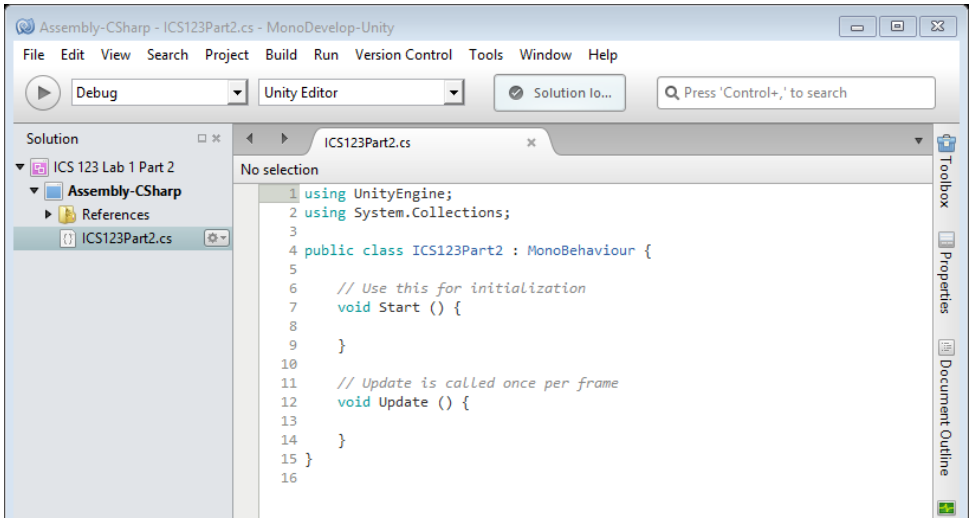

Figure 16: The 'empty' script file loaded in MonoDevelop

Notice that the script file you created isn't totally empty. It includes some namespace declarations (similar to import in Java), the class declaration (and you'll notice in C# that an inheritance relationship is represented by a *colon* instead of the keyword *extends* like in Java), as well as skeleton code for the two methods discussed in the background for Task 2. Note: You can ignore any warning about line endings.

We'll get familiar with the MonoDevelop IDE as we progress through the course. For now, here are two 'tidbits' about this editor:

- The window on the left is the **Solution View**. A Project in MonoDevelop is known as a Solution. In the Solution view, you will see the files that make up the Solution as well as some default namespaces. You'll notice that your Script has an extension *cs* which stands for C-Sharp.
- The *play button* in the upper left corner is for running the code independent of Unity. *Don't use this button!!!* Since we'll always being coding script components for Unity in this course, we'll use the *play button* in Unity.

6. Now that you have created your script component, you need an object to attach it to. Go back to the Visual Editor and go up to the *GameObject* menu item. Select *Create Empty*. A few things will happen: GameObject appears in the Hierarchy Tab and the Inspector Tab will show the Transform component of this new object (remember this is a default component that all game objects have).

7. *To attach your script to this object simply drag your script in the Project Tab and drop it on to the GameObject in the Hierarchy Tab*. Now your script is linked to the object! You can verify it by looking at the Inspector Tab and seeing if your script is now a component of the object (Figure 17 next page):

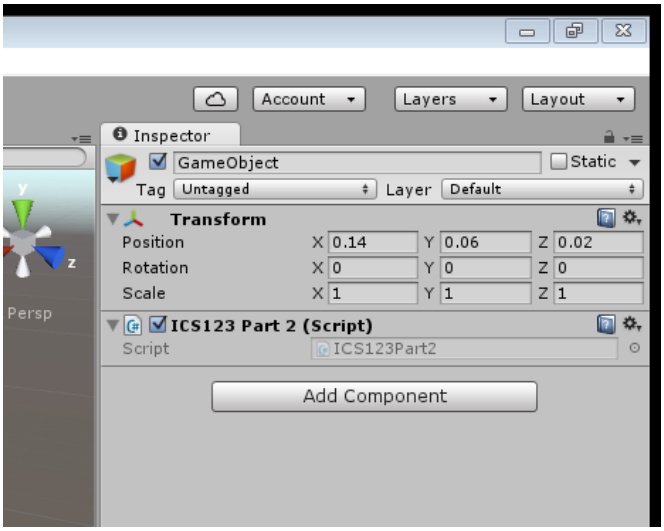

Figure 17: The added Script Component shown in the Inspector Tab

8. Your script won't do anything yet since we haven't added any code yet! Go back to MonoDevelop (remember you can double left-click on your script icon in the Project Tab inside the Visual Editor to open it in MonoDevelop if you previously closed it) and place the following line of code inside the `Start()` method:

```
Debug.Log ("My first Unity Project for ICS 123!");
```

   The `Debug.Log()` method prints the String in its argument to the Console Tab in Unity.

9. Now close MonoDevelop. It will ask you if you want to Save your changes. You definitely do!! Go back to the Visual Editor in Unity. Switch to the Console Tab at the bottom. Now hit the *play* button. You should see your message printed in the Console!!!

Submission:

Demo Task 2 Part 9 above to the Lab Instructor: • **Demo should show message printed in console tab when the Scene is played**

- A successful demo scores you 5 marks.