Lab 1 – Hello, Game Dev!

Prior Work: Unity & Visual Studio loaded to your system.

Resources: Hour 8 & 9 (pg 120+) in Tristem & Geig's *Unity Game Development in 24 Hours*

Or other C# Scripting guides (see https://docs.unity3d.com/ScriptReference/KeyCode.html and https://docs.unity3d.com/ScriptReference/Input.GetKey.html)

Objectives:

1) Getting started; adding assets; opening your IDE from within Unity
2) Opening Visual Studio from within Unity and start coding
3) Demonstrate ability to output "hello world" to the console (It's a tradition.)
4) Implement the Number Guessing Game as a console game

Opening Steps:

Open Unity. A splash page will open.

With **Projects** selected, click **New**.

Name your first project [firstname_lastname]_Lab1. Mine would be **Kate_Goodman_Lab1**

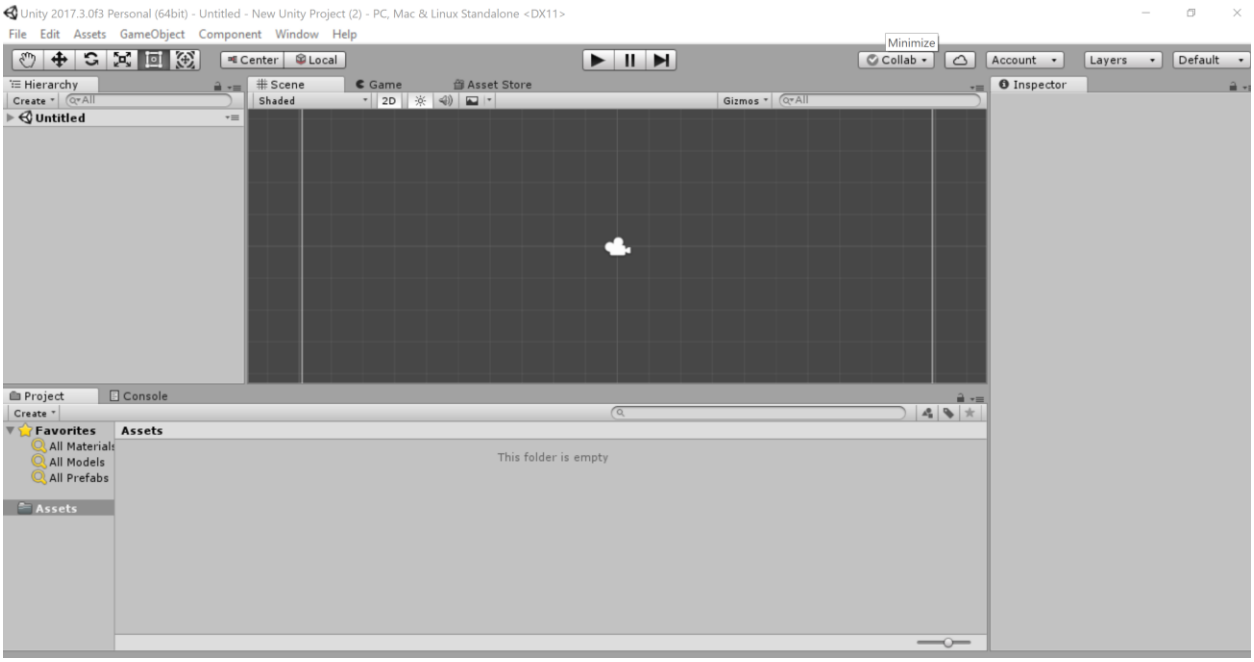
Select **2D** Game.

Choose a location to save the project.

Remember if you're using an Inworks laptop, to set this either to the desktop with the intent to move it OR to a thumbdrive location

Click **Create Project**.

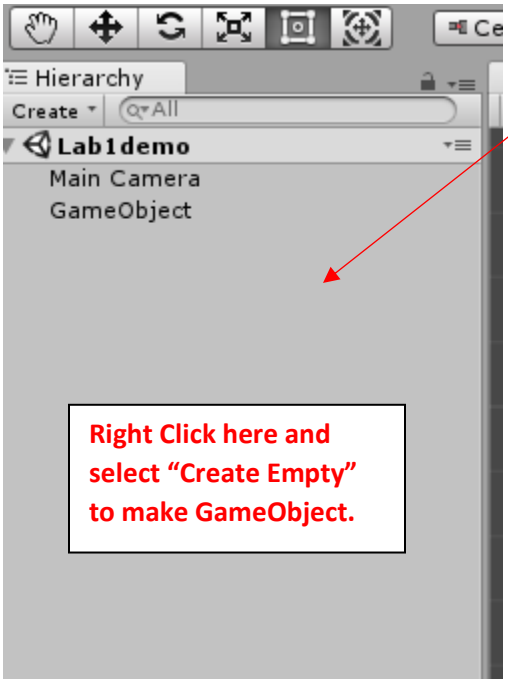
Unity will open with the windows looking something like this:



Let's change a setting that will help us see when we're in play mode and when we're not. Follow the directions at: (https://unity3d.com/learn/tutorials/topics/tips/play-mode-editor-tint?playlist=17114)

Creating a C# Script:

In the Assets window, right click. On that menu, select Create→ C# Script. An icon will appear in the Assets window, *NewBehaviorScript.cs.* Change this name to *NumberGuessingGame* (it should automatically retain the .cs extension). Now we need to associate this with a Game Object in Unity.



Under Hierarchy, right click and Create Empty.

This will populate the Inspector (right hand side) with attributes of this new, empty game object. Click and drag the icon for the script over to the Inspector tab and it will associate the script with the GameObject.

Now double-Click the icon for the script (over in the Assets tab at the bottom of the screen). Visual Studio should open (or the

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using System;
5
6
7
8   public class test : MonoBehaviour
9   {
10
11      // Use this for initialization
12      void Start()
13      {
14      }
15
16      // Update is called once per frame
17      void Update()
```

IDE you have linked to Unity). It will have default code that looks something like this. Yours will be whatever you named your script here.

Note:

Whatever we put in Start will only run once – when the game first opens. Whatever we put in Update will run once per frame of the game.

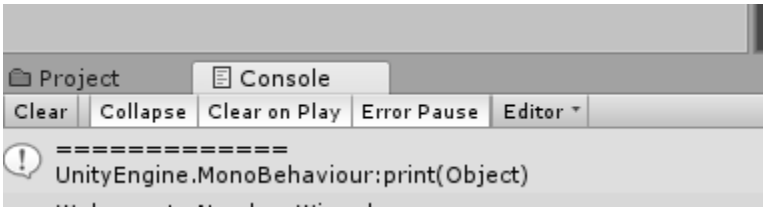
What Text do we want to see as soon as the game opens?

Let's try the traditional "Hello, World." Add the line inside Start:

```
void Start () {
            print("Hello World");
    }
```

Save within Visual Studio (shortcut is Ctrl-S), go back to Unity, and hit the Play button.

Your screen should shift to the color you set earlier, and the Console should say Hello World.
Where is your console? In Default layout, it will be a tab on the lower left, with Projects:



You can move this to wherever you want. I often put it on the right (vertical) area with the Inspector Tab. Feel free to click and drag it to a location that is convenient for you.
Click Play again to stop. The screen returns to the regular color.

What happens when you put a second print statement into Start?

What happens when you put a print statement into Update?

Change the script a couple times, and get used to going back and forth between Visual Studio and Unity.

Let's make a Number Guessing Game

Your goal is to make a one-player game, where the computer guesses a number the player is thinking of.

You want to control the range of numbers that can be guessed. For instance, you might set the range as 1-1000.

We're going to need some variables to hold our maximum and minimum guesses, and long with the current guess. I'm going to force my players to use whole numbers (integers), so I'm going to use:

```
// Use this for initialization
	int max;
	int min;
	int guess;
```

We need these above Start. (Why?)
We are declaring these variables (integers) without setting them to anything yet.

We're also going to add a new method between Start and Update called StartGame. This is so a game can start again without exiting the program. Remember, Start is only called once. StartGame could be called whenever we want. Now we have:

```
using UnityEngine;
using System.Collections;
//based upon an exercise by Tristem & Geig

public class NumberGuessingGame : MonoBehaviour {
	// Use this for initialization
	int max;
	int min;
	int guess;

	void Start () {
		print ("Welcome to the Number Guessing Game");
		StartGame();
	}
	// Method to restart Game within the program
	void StartGame () {


	}

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

	}
}
```

Let's set our three integer variables:

```
max = 1000;
min = 1;
guess = 500;
```

Let's put this inside StartGame so that they will be reset if we begin again.

Now write a set of print lines with directions for our player. We could write:

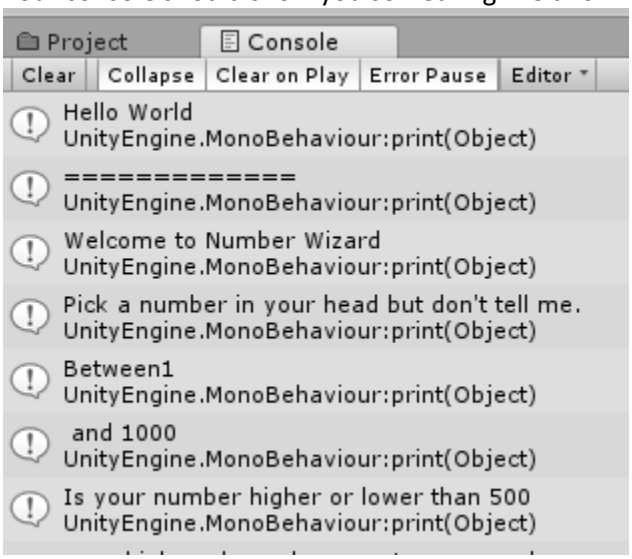
```
print ("The highest number you can pick is 1000");
```

But if we ever wanted to change the high end of the range, we would have to change it **both** in this statement and in how we set max. Instead we can write:

```
print ("The highest number you can pick is " + max);
```

Go ahead and write all the directions (you can use multiple print statements). Save the script and run in Unity.
Do you get all your directions to appear? Does it include your max and min variables in the directions?

Your console should show you something like this:



Notice that my line for the minimum says "Between1" with no space. How do we correct that?

Throughout this process, save your script, go back to Unity and hit play to see if it works as you imagined it would.

<u>Adding Player Responses:</u>

In Update, we're going to look for a key press.

We do this in C#:

```
(Input.GetKeyDown(KeyCode.UpArrow))
```

What if I don't want to use the up arrow? (Look it up! See links at start of Lab.)

This reads whether the key is pressed, but doesn't tell the game what to do. Let's add it to an if statement to our Update method:

```
if (Input.GetKeyDown(KeyCode.UpArrow)) {
                    print ("My guess was too low? I'll try higher");
}
```

How do if-else statements work? Can you add an else for the other possibilities (guess was too high, guess was right)?

And how do we increment the guess? I'm going to split the difference each time:

```
guess = (max + min) / 2;
```

Then I'll need to print the new guess.

This is going to get messy if I must add the code for changing the computer's guess in every if-else. Let's make a separate method for that:
Add under Update (after its closing brace, as a new method):

```
void NextGuess() {
        guess = (max + min) / 2;
        print("Higher or lower than " + guess);
        print("up = higher, down=lower, return = equal");
        }
```

Going back to Update, I'm going to add to my If statement:

```
if (Input.GetKeyDown(KeyCode.UpArrow)) {
                    min = guess; //this resets my min to my guess which was too low
                    NextGuess();
             } else if (Input.GetKeyDown(KeyCode.DownArrow)) {
                    max = guess;  //this resets my min to my guess which was too high
                    NextGuess();
             } else if (Input.GetKeyDown(KeyCode.Return)) {
                    print("I won!");
```

Your final code should look something like:

```
using UnityEngine;
using System.Collections;
//based upon an exercise by Tristem & Geig

public class NumberGuessingGame : MonoBehaviour {
	// Use this for initialization
	int max;
	int min;
	int guess;

	void Start () {
		StartGame();
	}

	void StartGame () {
		max = 1000;
		min = 1;
		guess = 500;

		print ("========================");
		print ("Welcome to the Number Guessing Game");
		print ("Pick a number and I will guess it!”);

		print ("The highest number you can pick is " + max);
		print ("The lowest number you can pick it " + min);

		print ("Is the number higher or lower than " + guess);
		print ("Up = higher, down = lower, return = equal");

		max = max + 1;
	}

	// Update is called once per frame
	void Update () {
		if (Input.GetKeyDown(KeyCode.UpArrow)) {
			min = guess;
			NextGuess();
		} else if (Input.GetKeyDown(KeyCode.DownArrow)) {
			max = guess;
			NextGuess();
		} else if (Input.GetKeyDown(KeyCode.Return)) {
			print("I won!");
			}
	}

	void NextGuess () {
		guess = (max + min) / 2;
		print ("Higher or lower than " + guess);
		print ("Up = higher, down = lower, return = equal");
	}
}
```

Run this, make sure you understand it. Then see if you can answer the following:

Follow up questions

1. Explain what happens if you don't have the line:

   `max = max + 1;` (in the StartGame method)

2. Change the program so that `guess` starts as a random number.
3. Add a way for the game to begin again.
4. Add different areas to the code, trying out **while**, **switch**, **for**, the **?** conditional operator, **do-while** loop. These don't have to be central to the game – just demonstrate your understanding of the syntax.