Unity

Introduction

CSC 631/831
Spring '14

What is Unity?

- Game Engine
  - A complex system designed for the development of both 3D and 2D video games
  - Multiple components work together to bring a game to life:
    - Graphics Rendering Engine
    - Physics Engine
    - Scripting
    - Etc.

What is Unity? (cont'd.)

- Cross-Platform
  - PC & Mac (Windows, OS X, etc.)
  - Mobile Devices (iOS, Android, etc.)
  - Game Consoles (PlayStation, Xbox, etc.)
- Editor w/ Built-in IDE
  - C# using MonoDevelop
  - JavaScript / UnityScript (Not Used)

Game Engine Core

- Graphics Rendering Engine
  - Responsible for taking graphical data of models and generating a visual image to the screen
  - 3D to 2D conversion
- Physics Engine
  - Simulate physics to allow object interactions
  - Collision detection

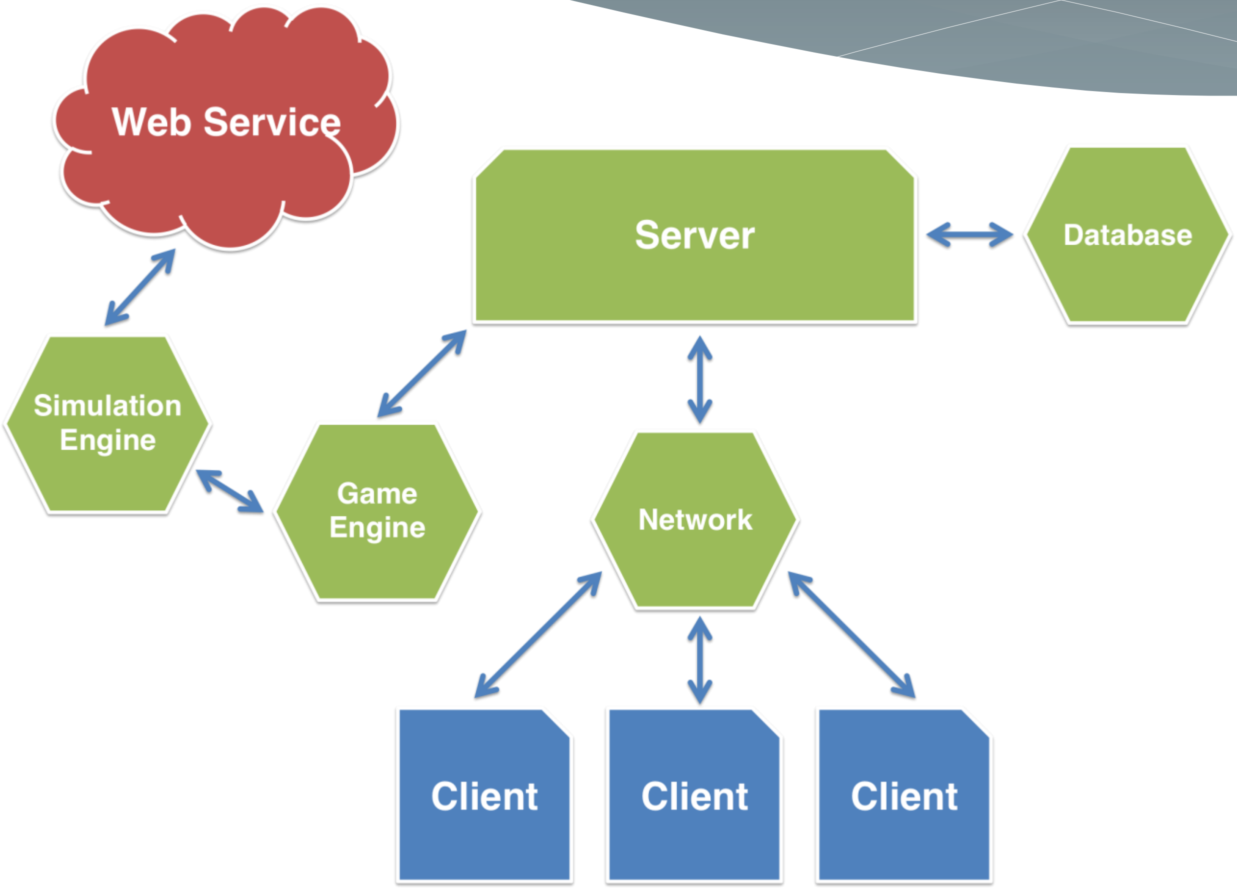
Game Engine Core (cont'd.)

- Scripting
  - Instructs the game engine to perform certain tasks
    - Respond to user inputs
    - Create and handle events
    - Control object behaviors
- Other
  - Sound, Animations, Networking and more

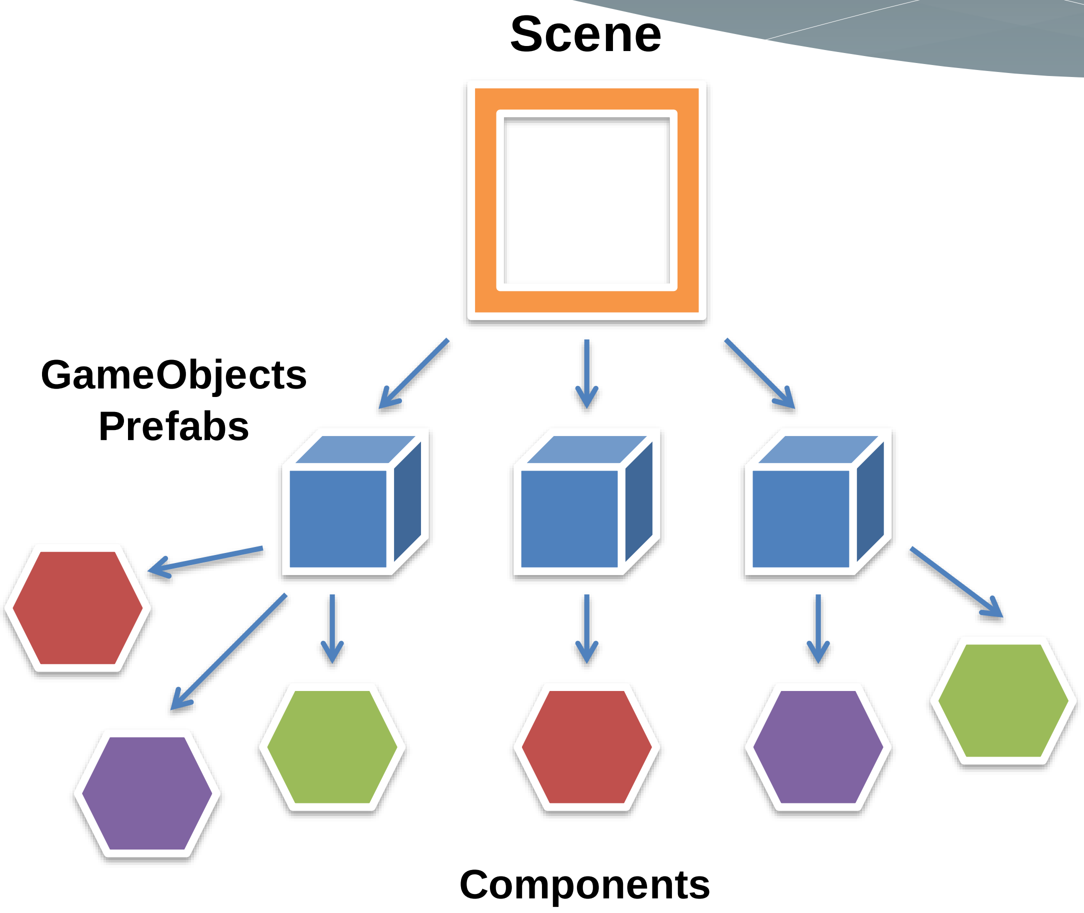
How will this be used?

- Unity is strictly used for development of the Game Client, which is 1 of 2 major components
  - Game Client will be required to connect to a Game Server, which will be discussed at a later point
- Game Client is responsible for:
  - Displaying visuals
  - Respond to user inputs
  - Act out the game logic

Architecture Design



Scene Hierarchy



Scene

- What is a scene?
  - "Level" container that contain objects for your game
    - GameObjects
    - Prefabs/Models
    - Scripts
    - Etc.
  - Can just only contain GUI menus as well
  - At least one is required to function properly

GameObject

- In its most basic form, its just an “empty” container
- Serves no use unless special properties called Components are added into it
- No GameObjects are truly empty because of the Transform component
  - Defines the position, rotation, and scale
- Every object in your game is a GameObject

Components

- Defines the behavior of every GameObject
- Gives GameObject purpose
- There are many different components
  - Transform
  - Camera
  - Scripts
  - Colliders
  - Etc.

Prefab

- Type of asset
- Reusable GameObject
- Unity creates an instance of it whenever added to the scene
- Modifying the components of a Prefab changes the instances
- Normally is made up of a model along with other components to be loaded multiple times

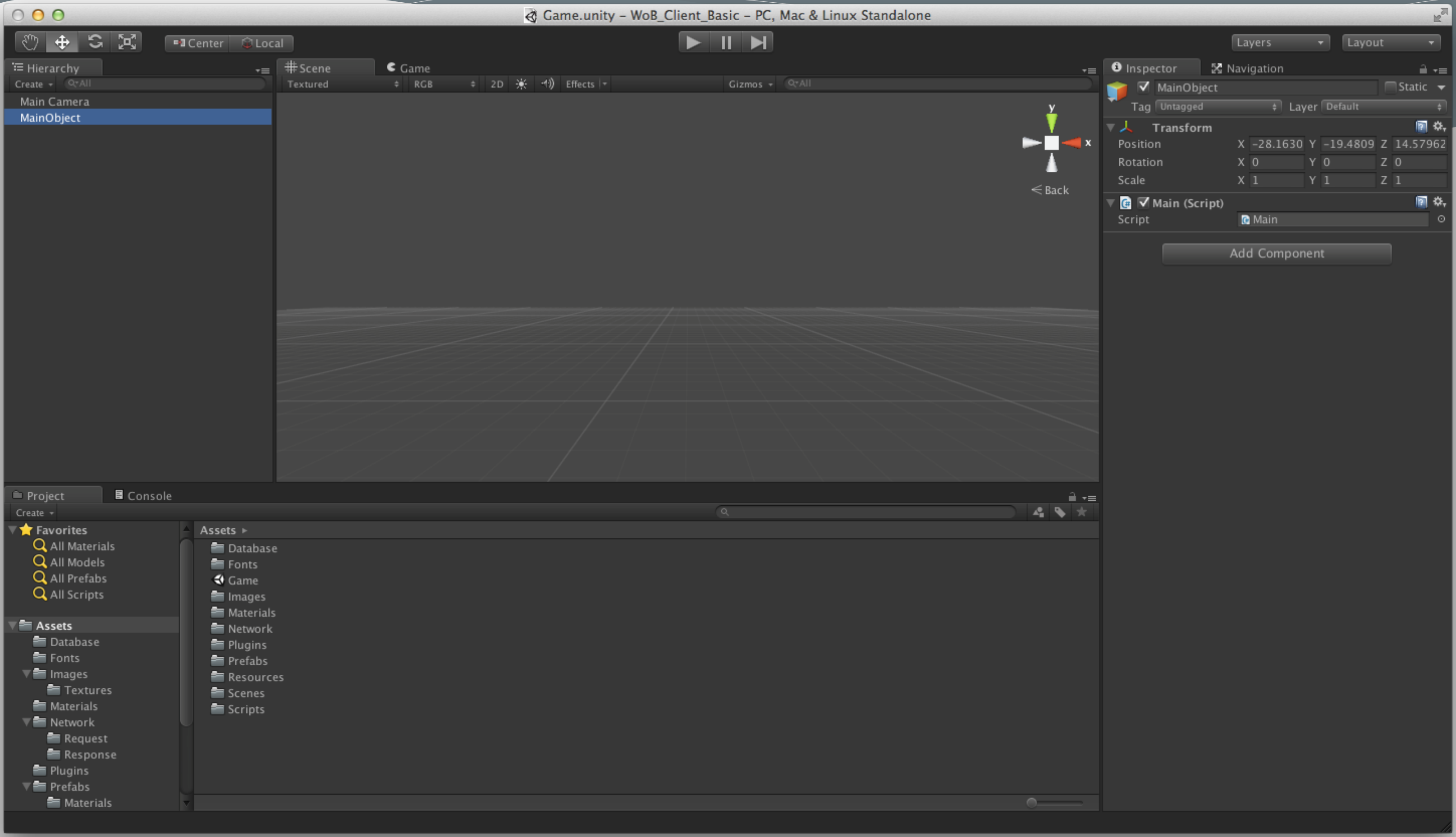
Prefab (cont'd.)

- Example:
  - Elephant Prefab
    - Transform
    - Collider
      - Collision purposes
    - Elephant Model
      - Materials/Textures
    - Scripts
      - AI, etc.

Script

- Adds another level of behavior in GameObjects
- Dictate game logic
- Every individual script is its own Component
- Written in C#
  - Also in JavaScript/UnityScript
    - Not in this class

Unity Editor



MainObject — Static
Tag: Untagged — Layer: Default

Transform
Position X -28.1630 Y -19.4809 Z 14.57962
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

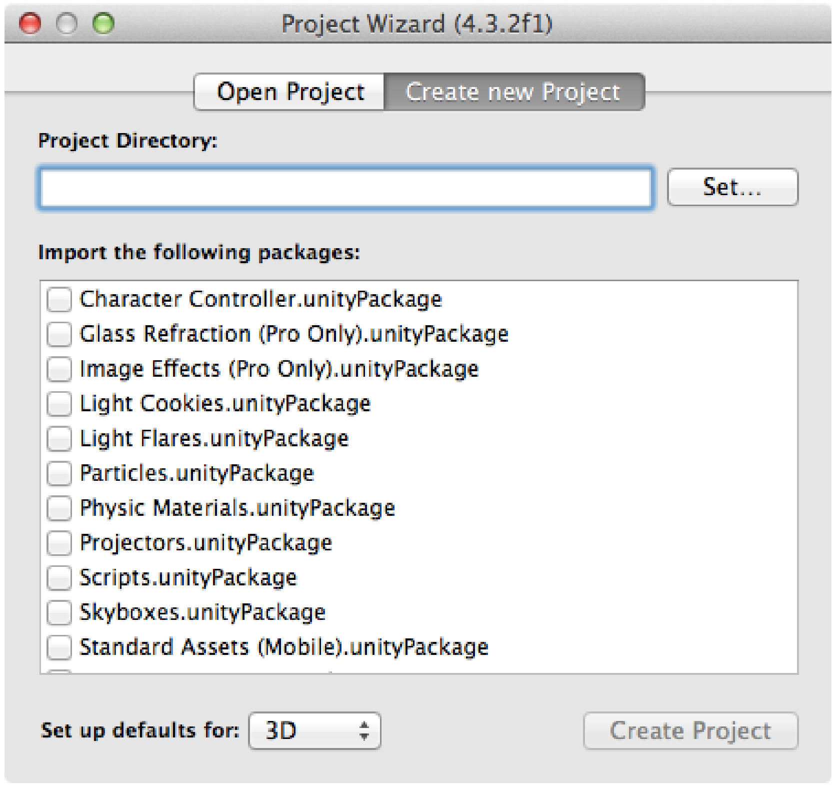
Main (Script)
Script: Main

Getting Started

- Download Unity
  - http://unity3d.com/unity/download
- Install Unity
  - Installation should be very straightforward
- If asked, please skip the 30 day trial of Unity Pro since it may come to use later in the semester, if needed

Getting Started (cont'd.)

- You'll need to create a new project
  - File > New Project
  - Skip the packages > Create Project



Project Wizard (4.3.2f1)

Project Directory:

Import the following packages:

- [ ] Character Controller.unityPackage
- [ ] Glass Refraction (Pro Only).unityPackage
- [ ] Image Effects (Pro Only).unityPackage
- [ ] Light Cookies.unityPackage
- [ ] Light Flares.unityPackage
- [ ] Particles.unityPackage
- [ ] Physic Materials.unityPackage
- [ ] Projectors.unityPackage
- [ ] Scripts.unityPackage
- [ ] Skyboxes.unityPackage
- [ ] Standard Assets (Mobile).unityPackage

Set up defaults for: 3D

Getting Started (cont'd.)

- Your first assignment will require you to open an existing project instead of creating a new one

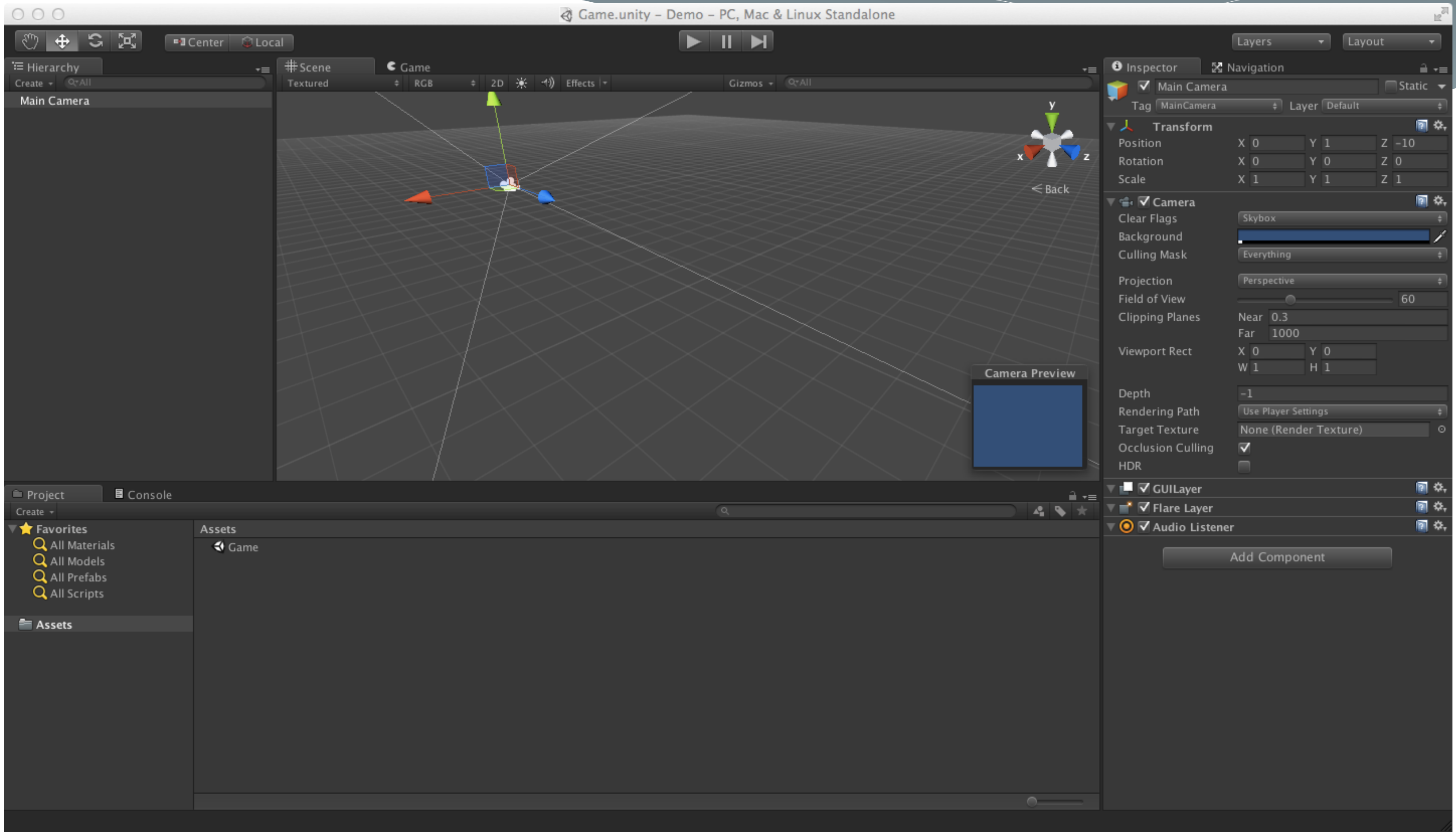
Hello World!

- Just a basic example to get started
- We’ll be doing the following:
  - Setting up a Scene
  - Adding a GameObject
  - Display “Hello World!” using a C# Script
    - Console
    - In-Game

Setting Up Scene

- By default, a scene is already created for you
  - Camera will always be the first object in the scene
- First thing you should do is save the current scene by giving it a name
- You'll need to create new scenes whenever you need to switch from one to another
- For example:
  - Switch from Login to Level

Setting Up Scene (cont'd.)



Main Camera — Static
Tag: MainCamera — Layer: Default

Transform

Position	X 0	Y 1	Z -10
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Camera ✓

Clear Flags	Skybox
Background	🟦
Culling Mask	Everything
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	-1
Rendering Path	Use Player Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	

✓ GUILayer
✓ Flare Layer
✓ Audio Listener

Adding GameObject

- For your game to do anything, you need a Script, but before you can add one, you'll also need a GameObject attached to the Scene
- When your game runs, Unity will reach out to all GameObjects attached to the Scene
- As it goes through your GameObjects, it'll pick up all its Components as well as any Scripts attached to it

Adding GameObject (cont'd.)

- In the menu, select GameObject > Create Empty
- A GameObject named "GameObject" will be inserted into your scene
- For this demonstration, this GameObject will need to attach a script component either by:
  - Selecting GameObject > Add Component
  - Dragging a pre-existing script from the Project tab to the GameObject

What is C#?

- Programming language similar to Java
- If you know Java, C# is really easy to learn
- Shares almost the exact syntax
- Few exceptions when using C# with Unity:
  - Component-based scripting doesn't follow the traditional use of constructors
  - Namespaces are not used in Unity

What is C#? (cont'd.)

- MonoDevelop provided by Unity is one way to code for it
- For Windows users, MS Visual Studio is another option if you don't prefer the built-in IDE

Hello World! Script

- Once a script is created, you'll need to open it with Unity's built-in IDE called MonoDevelop

```
using UnityEngine;
using System.Collections;

public class HelloWorld : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

MonoBehaviour Class

- MonoBehaviour
  - All C# classes requiring Unity methods will have to extend from this
  - Required whenever a script is treated as a component
  - Contains important methods such as:
    - Start()
    - Update()
    - Also, Awake() and others

Awake() and Start() Methods

- Awake() Method
  - Script components do not use constructors
  - This method is called by Unity to initialize variables before running, similar to a constructor
- Start() Method
  - Similar to Awake() except that it'll run once the engine kicks in
  - In most cases, you can just use Start() over Awake()

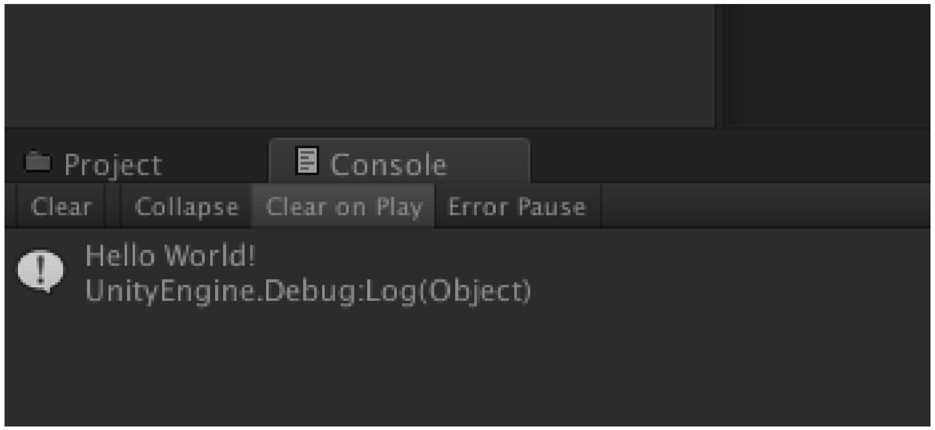
Update() Method

- Update() Method
  - Once the engine is running, any methods named Update() will be called once per frame
  - This is where most of your logic code belongs
  - For example:
    - Character needs to move along a path
    - Your Update() method will include code to make it walk a short distance based on speed

Output “Hello World!” (Once)

```
using UnityEngine;
using System.Collections;

public class HelloWorld : MonoBehaviour {

    // Use this for initialization
    void Start () {
        Debug.Log("Hello World!");
    }

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

    }
}
```

Output “Hello World!” (Once)



```
Hello World!
UnityEngine.Debug:Log(Object)
```

Debug.Log() Method

- Common method used to output strings to Unity Editor's console
- This is a Unity method, which is similar to C#'s Console.WriteLine()
- Similar to System.out.println() from Java

Output “Hello World!” (Multiple)

```
using UnityEngine;
using System.Collections;

public class HelloWorld : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        Debug.Log("Hello World!");
    }
}
```

Display “Hello World!”

```
using UnityEngine;
using System.Collections;

public class HelloWorld : MonoBehaviour {

    // Use this for initialization
    void Start () {
        Debug.Log("Hello World!");
    }

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

    }

    void OnGUI () {
        GUI.Label(new Rect(30, 30, 100, 100), "Hello World!");
    }
}
```

OnGUI() Method

- OnGUI() Method
  - Similar to Update() method, which is called once per frame
  - This method is reserved for drawing GUI elements on the screen once your game is running
  - GUI Elements:
    - Windows
    - Boxes
    - Labels, Etc.

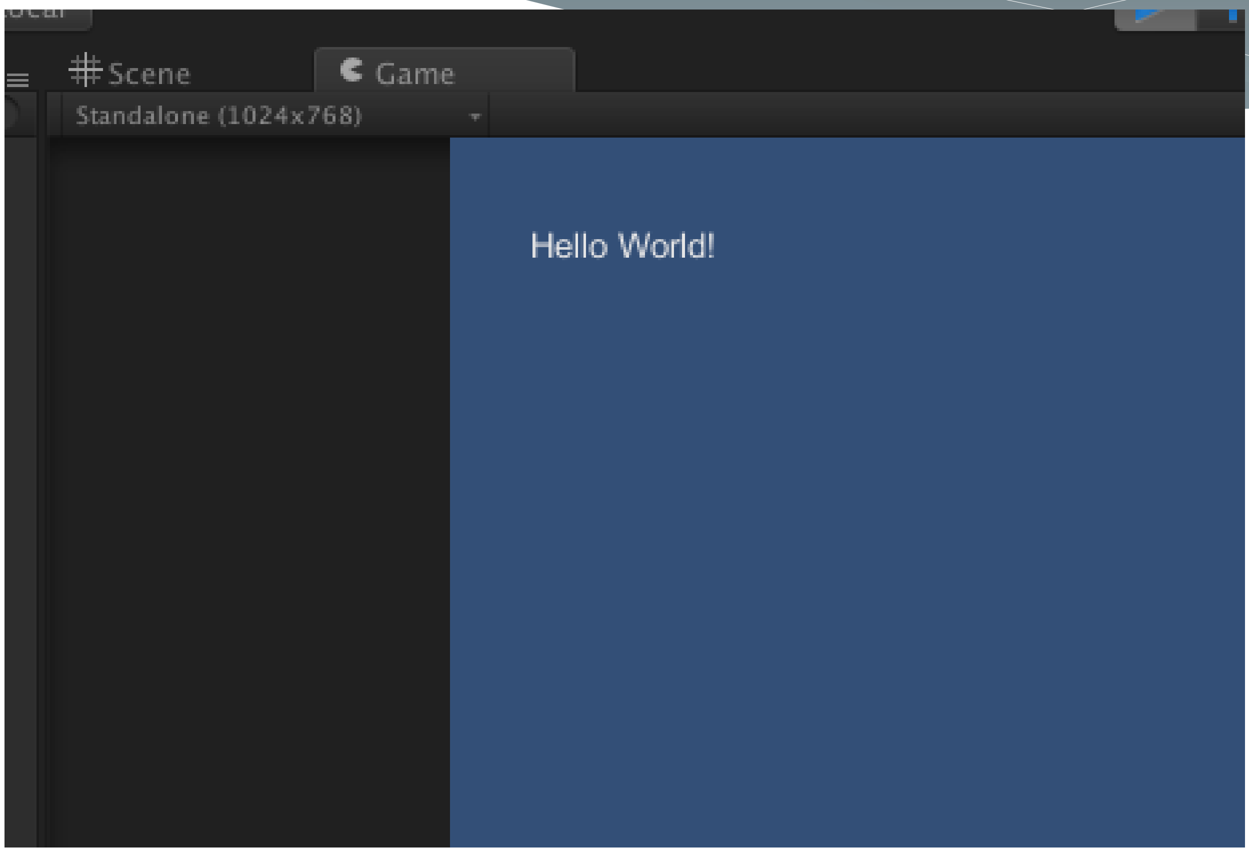
GUI Methods

- There’s a method for every GUI element you’d like to create whether that’s for a box, window, label, etc.
- One Example:
  - GUI.Label()
    - Used to draw a simple line of text on the screen

Rect (Rectangle)

- Every GUI element requires use of a Rect structure using the following constructor:
  - Rect(x, y, width, height)
- The rectangle must be large enough to display its contents otherwise there will be a cutoff.

Display "Hello World!"



Hello World!

Online Resources

- Unity User Manual
  - http://docs.unity3d.com/Documentation/Manual/index.html
- Unity Scripting Reference
  - http://docs.unity3d.com/Documentation/ScriptReference/index.html