Spring 2021

CSE 190

VR Technologies

Discussion 2

Guowei Yang

UCSDCSE

ANNOUNCEMENTS

- Homework 1: Whack-A-Mutant **DUE THIS WEEKEND**
  - **Due April 18th @ 11:59PM**
  - **START NOWWWWW!**
- Extra Credit Opportunities
  - Sound Effects (whack and miss, whack and hit, time expired)
  - Wiggle the Virus!
  - VR Support

UCSDCSE

AGENDA

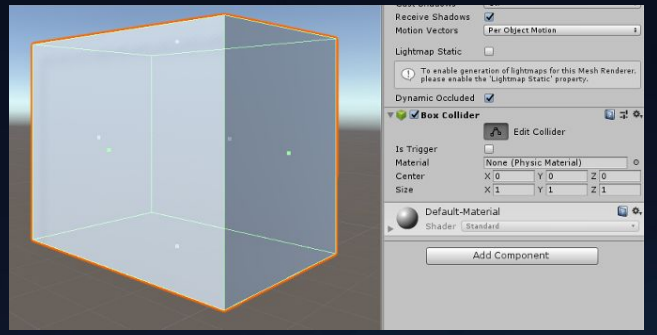
- Collision Detection
- C# Programming

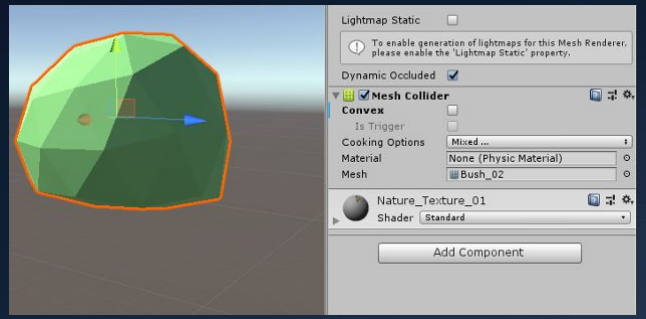
UCSDCSE

COLLISION DETECTION

Collision Detection

- Handles collision for you with components
- Game objects created will have it enabled by default
- Use the correct type for your application
- Box versus Mesh Colliders
    - Tradeoff between Performance vs Accuracy
- Necessary for a gaze system in Unity



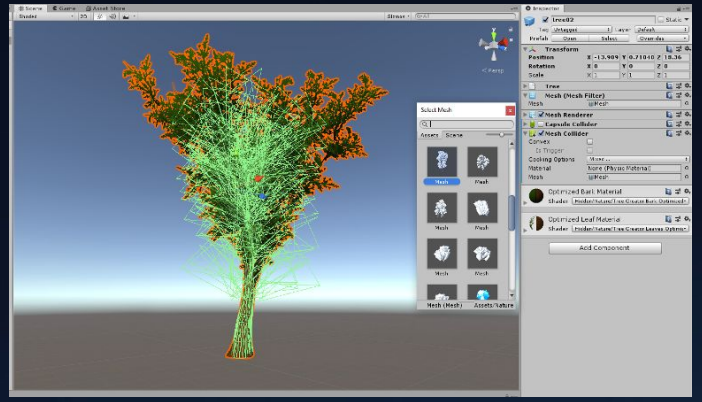


Box Collider

- Simple
- Fast
  - Wraps a bounding geometry over the GameObject
  - Doesn't have to check very many planes
- Tradeoff
  - Accuracy, not suited for high precision and can be clunky in a dense environment

Mesh Collider

- Complex
  - Uses object's mesh to create the collider
- Only scans one side of the mesh
  - Will only trigger when going through one way
- Computationally Heavy
  - Has to check every single face
- Tradeoff
  - Performance, accurate and well suited for precise and authentic hits but should be used sparingly



C# Programming

C# Intro

- Created in 2000 by Microsoft
  - Uses C/C++ as a base but is very similar to Java
  - Documentation :

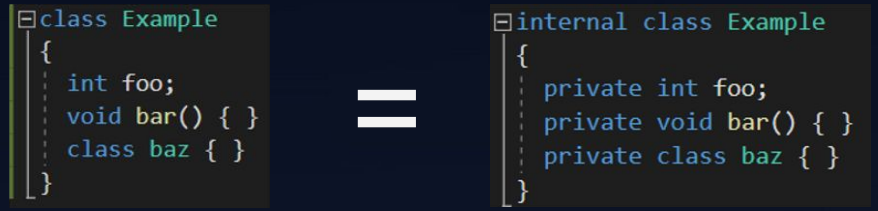
    https://docs.microsoft.com/en-us/dotnet/csharp/
- Scripts written in Unity are in C#
  - Compiles upon detecting file change
  - All scripts must compile prior to build

Language Basics

- Primitive Types
  - `int` / `bool` / `char` / `float` / `double` / `short` / `long` / etc...
  - Are actually **objects**, they are ValueType class derivatives
- Namespaces
  - Highest level classification, used to group classes and below
  - Unity imports these by default:
    - `using System.Collections;`
    - `using System.Collections.Generic;`
    - `using UnityEngine;`
- C / C++: `#include <stdio.h>`
- Java: `import java.util.*;`

Access Levels / Visibility

- If not specified, everything takes the most restricted modifier



- Need to set variables to public to access in Unity and outside of script
- Creating objects
  - `public Object obj = new Object();`
  - `public Object objRef = obj;`

Basic Syntax

Loops

- While: `while (true) { … }`
- DoWhile: `do { … } while (true);`
- For: `for (int x = 0; x < 10; x++) { … }`
- For Each: `foreach (int x in array) { … }`

Keyword: `ref` – Reference, Same as & in C / C++

- `foreach (ref int x in array) { … }`
- `void foo(ref int bar) { … }`

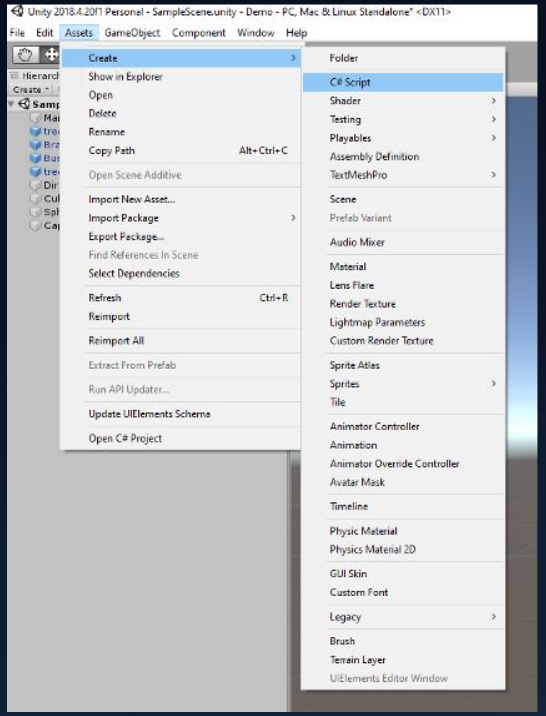
Keyword: out – Similar to passing by reference, specifies an output variable

- `void foo(out int bar) { … }` // bar has to be set in the function

Keyword: var – Similar to auto in C / C++, implicit type definition

- `var x = 10;` // x is of type Int32

Creating C# Script



Scripting

Script classes are global
- Shared between all scripts, declare / define them like a normal object

```
public Script script = new Script();
```

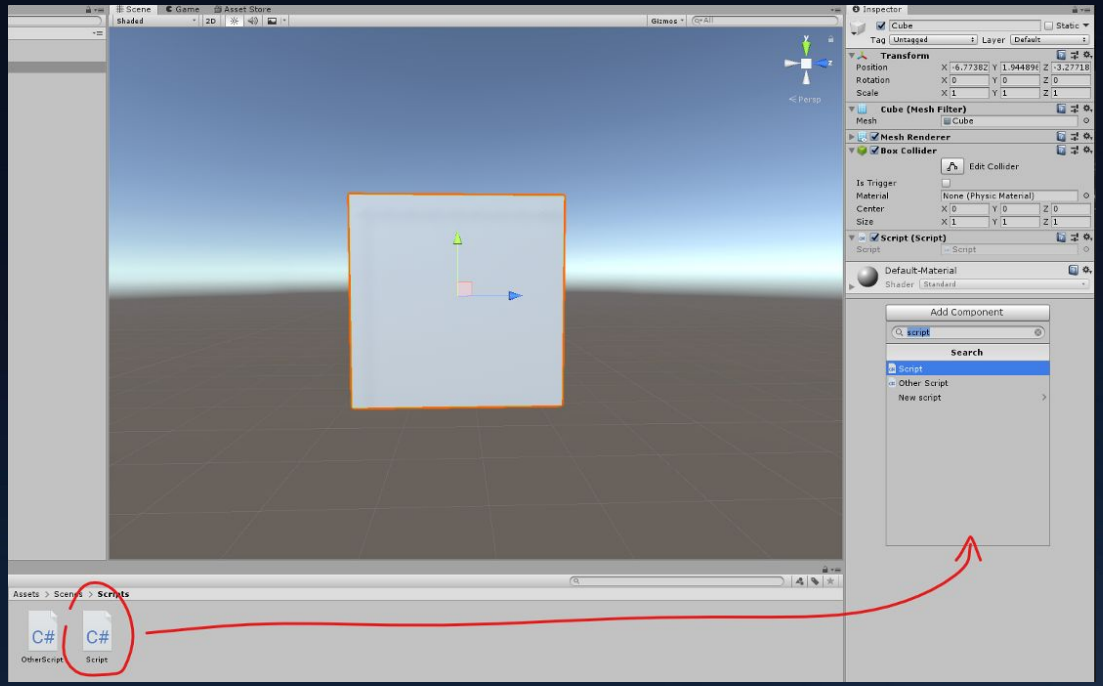
GameObjects are objects
- Declare / Define them like so

```
public GameObject gameObj = new GameObject();
```

Scripts are components
- Access them from a game object via `GetComponent<Component>()`

```
gameObj.GetComponent<ScriptClass>();
```

Bind to Objects



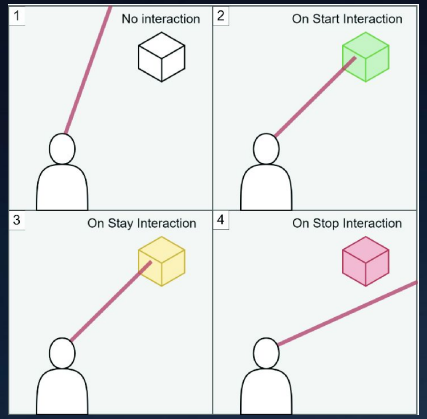
Scripting - Gaze System

Form of interacting with the environment

- Most commonly found in mobile, less so on other platforms with dedicated controllers
- Usually will have some kind of reticle

Easy to implement in Unity

- Ray – Creates a ray with specified position and direction
- Can retrieve information with what it collides with (colliders on GameObjects)



Scripting - Gaze System

`void ProcessGaze()`

- `Ray gazeRay = new Ray(transform.position, transform.forward);`
  - Ray is part of UnityEngine, transform is the current GameObject's (Camera's)
- `Physics.Raycast(gazeRay, out hitInfo)`
  - Determines if something was hit and returns a bool, information stored in hitInfo
- `GameObject hitObject = hitInfo.collider.gameObject;`
  - Pulls a reference to the hit GameObject in hitObject
- `GazeableObject gazeObj = hitObject.GetComponent<GazeableObject>();`
  - If a GameObject with the GazeableObject script was hit, set the reference to it

QUESTIONS?