XR LIBRARIES

INSTITUTE of Museum and Library SERVICES

VR

An introduction to VR programming

VR Overview

- Summary, Objective, Materials
- Essential Questions
- Unit Structure
- Unit Content Overview
- Materials and platforms
- Key Features
- Final Project
- How to Use this Guide

Lesson 1 Introduction to the course

Introduction to the course goals, key features, expectations

Lesson 2 Demonstration Project

A demo project to show more advanced Unity features

Lesson 3 Game Design Document

Plan our work before we do it.

Lesson 4 Minimum Viable Product

What is the least amount of product needed

Lesson 5 Introduction to Scripting

How to create custom features for our game.

Lesson 6 3D Modeling

Create 3D Models with VR tools

Final Lesson Presentation and Extended Goals

Show off the completed work and make plans for extending it.

Appendixes

C# for Unity Cheat Sheet

Modeling Cheat Sheet

Glossary of words

Resource Page

SUMMARY:

- This course teach students how to make a simple complete game using the Unity3D game engine. It encourages independent use of other information resources provided in the library such as software training available on Lynda.com.

OBJECTIVE

- The goal is to promote learning through creative engagement and give students the skills and confidence to develop VR programs or continue with more advanced classes.

MATERIALS NEEDED:

- Enough computers for the students. If hardware is limited, students can pair up to create their project.
- At least one computer with a VR headset attached. The more headsets available the better.
- Unity3D installed on all computers that will be used for development.
- Flash Drive, Cloud Storage or Network Connection.
- A way to move the projects from development computers to VR computers

Essential Questions

What is the goal of this course?
This course teach students how to make a simple, but complete, interactive experience using free software and tutorials.

Who can teach it?
Anyone with a firm grasp of computer use. It does not require a degree in computer science or mathematics, but does require a curious mind and willingness to learn. This course expects teachers to have a firm grasp on Unity through practice and patience.

How long does it take to learn to use Unity?
The very basics of Unity can be picked up in as little as an hour, but it is recommended that teachers spend at least 10 hours working with Unity before trying to teach it to others. This prepares you to solve the most common problems that arise when students are first learning.

Where can I get help learning Unity?
The Unity website has a good array of written documentation and video tutorials. There are also thousands of hours of video tutorials in places like YouTube, Lynda.com, Udemy and others. Links to helpful tutorials are provided before each lesson and are also available in the online resources section for this lesson plan.

What do students need to know prior to the class?
Students should have basic computer skills and be comfortable navigating Windows and searching the Internet. The course is intended for those thirteen and up, but could be followed by anyone, depending on their level of experience.

Unit Structure

This course is designed to be run over many sessions totaling between 30 and 40 hours of instruction. There are 6 lessons designed to introduce concepts. Students should be given plenty of dedicated time to explore those concepts. For example, a session on 3D modeling should be followed up with a period of study and experimentation.

Unit Content Overview

Augmented and virtual reality(which are sometimes referred to collectively along with other emerging mediums as XR or eXtended Reality) are becoming mainstream. They have captured the imagination of the public for decades and are just now starting to reach a level of cost and quality that matches the expectations set by the books and films of years past.

Students will...

1. Learn about the current and potential future uses of this technology
2. Learn the basic terminology of virtual reality and game development
3. Create their own content, including code and 3D models
4. Explore the benefits and limitations of the medium

Final Project

Every lesson in this course helps students create individual parts of a final project. They will create a new project in the early lessons and continually add to is as the course progresses. The capstone of the course is to invite people to check out the completed work in a public demonstration event.

How to Use this Guide

As with any lesson, it is best to familiarize yourself with the complete lesson before teaching a class. There are many parts which require a working knowledge of software beyond what the typical user would encounter.
As with any lesson, the techniques presented here should be adapted to the instructor and the students' needs.

Appendix

This course is designed to be run over many sessions totaling between 30 and 40 hours of instruction. There are 12 lessons designed to introduce concepts. Students should be given plenty of dedicated time to explore those concepts. For example, a session on 3D modeling should be followed up with a period of study and experimentation.

These are examples of prompts you'll see throughout the course!

Game Development Journal

There are a few ideas that will be continually revisited throughout the course. The first is a Game Development Journal. This is where the student can write ideas, notes, pseudocode, scene layouts, 3D models and more. It should be a paper journal, but the format is flexible. A student may prefer an unlined sketchpad, graph paper or a composition book. It is recommended that each student has a journal that they bring to every session. Throughout this lesson plan, there are prompts for entries into the journal that look like the box below.

- This is a Game Development Journal prompt.

Game Development Journal

Question and Answer

To promote student engagement, the teacher should ask questions for the students to answer. These often have no single right answer and are designed to promote discussion. Throughout this lesson plan, there are prompts for entries into the journal that look like the box picture below.

- This is a question that you will ask students?

Questions and Answers

Activity in Unity

In each lesson, the students should be allowed ample time to implement what they learned through lecture and discussion. During this time, the instructor should be walking around answering specific questions and offering advice related to Throughout this lesson plan, there are prompts for entries into the journal that look like the box picture

- This is an activity that students will complete in Unity.

Unity Activity

Lesson 1:

Introduction to the course goals, key features, expectations

Overview

Students are introduced to the course, expectations, each other and the main piece of software they will be using, Unity3D. This is an opportunity to do housekeeping as far as registration and attendance.

Learning Objectives	**Students will:** 1. Meet other students and get to know them 2. Memorize the steps needed to open projects and create new scenes 3. Recall the basic keyboard and mouse controls for navigating the scene view 4. Recall the steps needed to create objects in Unity 5. Relate the size of real world objects to the units used in Unity(meters) 6. Apply knowledge of 2D coordinates to virtual 3D space 7. Draw two orthographic views of a floor plan 8. Produce a 3D virtual representation of the orthographic views
Materials	1. Game Design Journals. Any sort of notebook, sketchbook or binder that the students can keep. 2. Unity example project loaded on all computers 3. One copy (2 sides) of the Unity Interface Cheat Sheet found in the appendices
Links	**A list of improv style icebreaker games** https://medium.com/@astarteny/team-building-with-improv-games-c63ae0e02661 **Video tutorial of the Unity interface** https://unity3d.com/learn/tutorials/topics/interface-essentials/interface-overview **Overview of the Unity interface** https://unity3d.com/learn/tutorials/topics/interface-essentials/interface-overview
Suggested Time	2 hours
Grouping	Single students
Vocabulary	-Unity3D -Virtual Reality -Scene -Project -VRTK -SDK -Primitives -Game Design Document -Game Development Journal -Play Mode -Edit Mode -Inspector

Preparation

- Make sure Unity3D is installed on all development PCs.
- If headsets are attached, make sure they are functional.
- Create a test project before class to ensure all hardware and software is working properly.

Procedure

1. Lead an icebreaker you are comfortable with that introduces the students to each other and facilitates communication. There is a list of icebreakers in the links section.

2. Ask students some of the following questions.

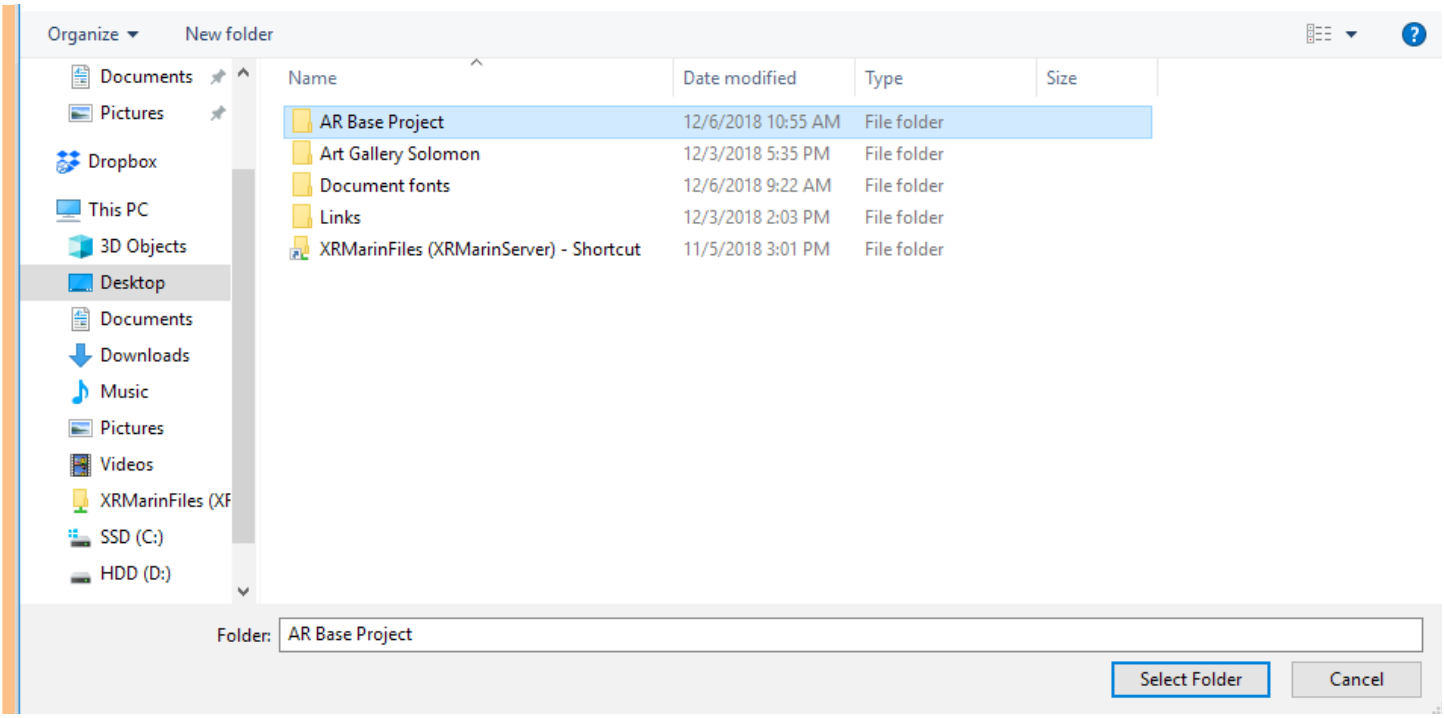
- What is Virtual Reality?
- Have you ever used VR before?
- The current focus of the VR market is gaming. What else can VR be used for?
- Do you think the market for VR is expanding, why or why not?

Questions and Answers

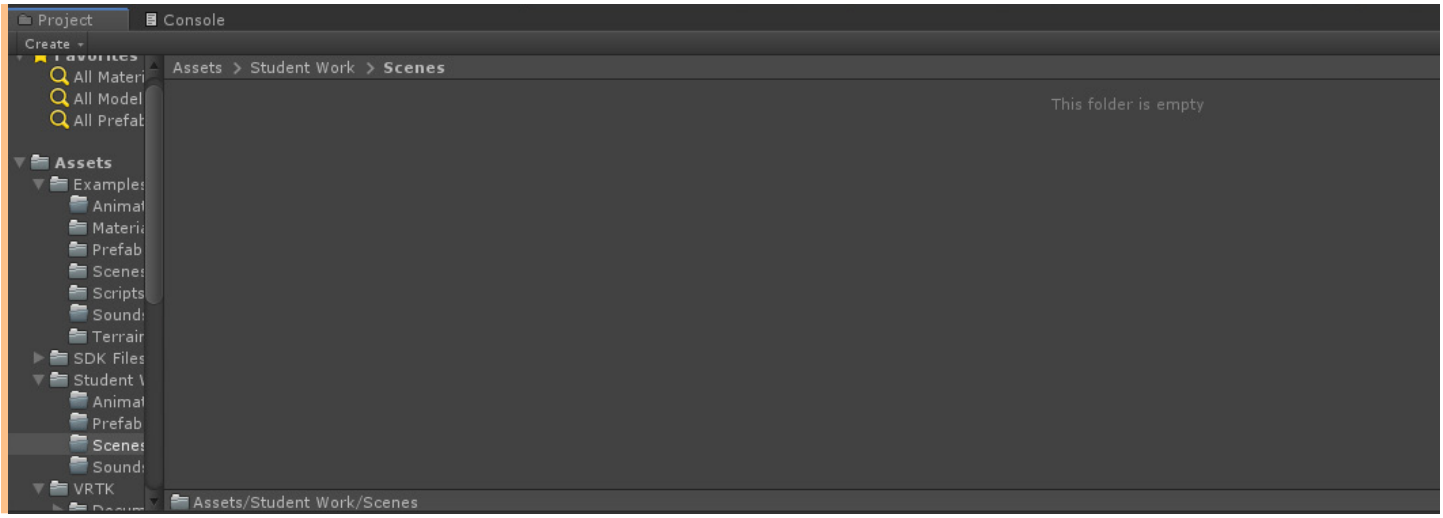
3. Distribute copies of the Unity Interface Cheat Sheet, found in the appendices.

4. Have students access the demo project by running Unity and clicking Open.

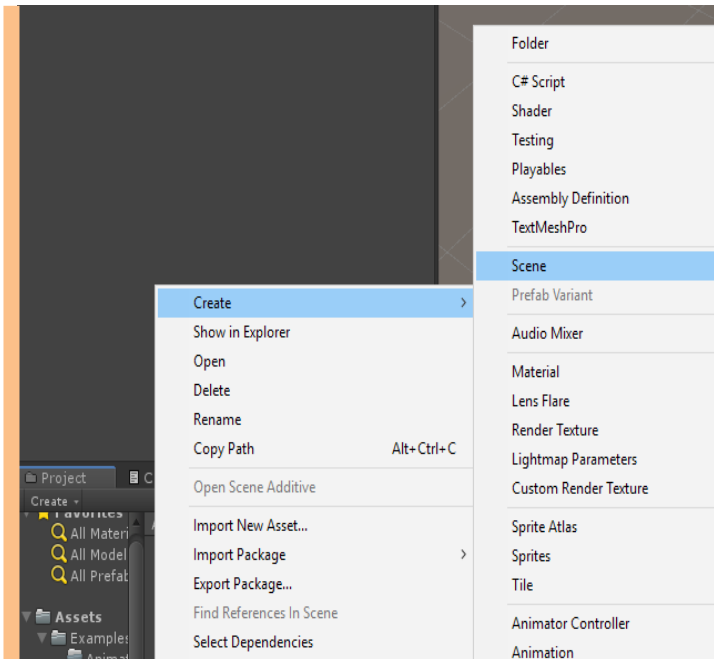
5. Have students navigate to the location you saved the project, select the VR Base Project folder and click Select Folder.



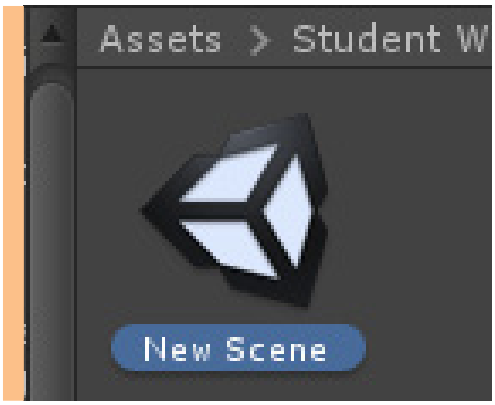
5. Once the project is loaded, have students use the Project Window to navigate to Assets>Student Work>Scenes



6. Instruct students to create a new scene by right-clicking in the Project Window and selecting Create>Scene



7. The students can rename the scene. Once they have named it, instruct them to double -click on the scene file in the Project Window



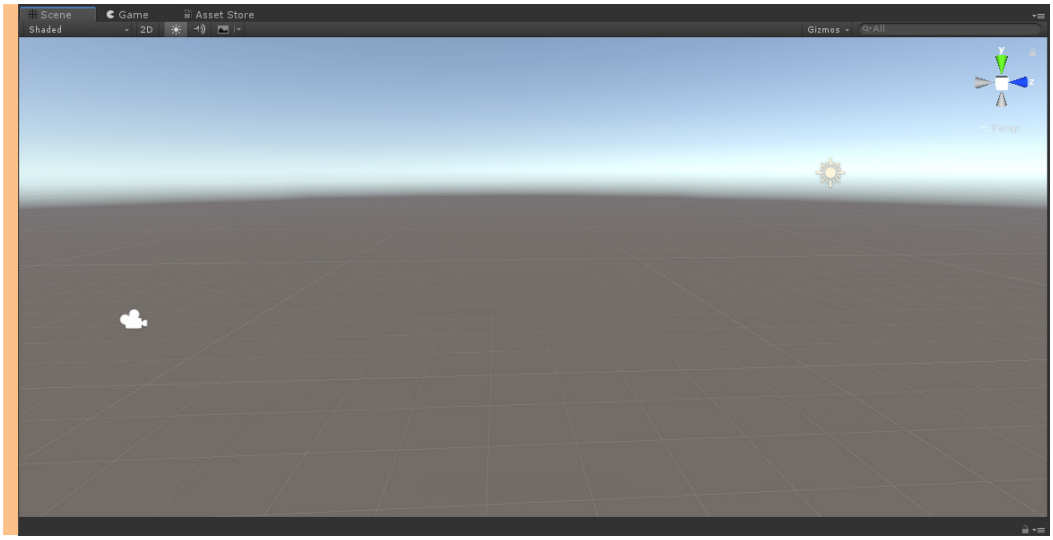
8. Tell the students that this is a blank scene. Show the two default objects in the Hierarchy window; the Main Camera and the Directional Light. Explain that these objects will added to any new scene.

9. Discuss the concept of scenes as containers that hold artwork, code and other assets. Use the analogy of levels or stages in a game.

- What do you think a scene is?
- Why would a developer want to use different scenes? Why not just have everything in one scene?

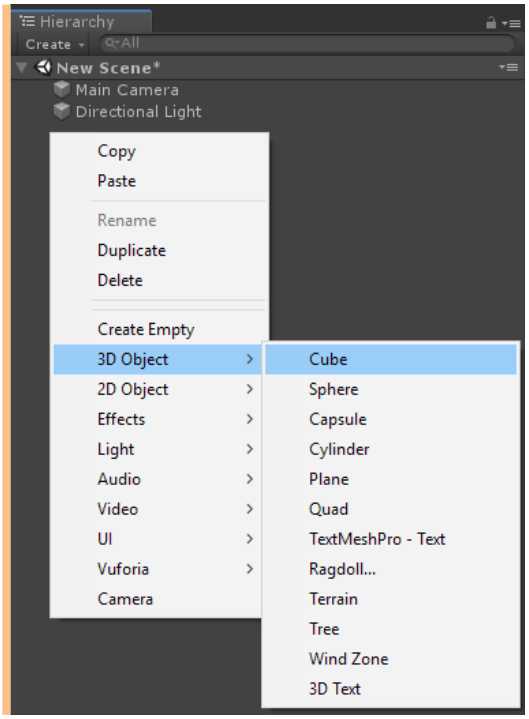
Questions and Answers

10. Explain the Scene View. Point out that every object in the Hierarchy Window, inlcuding the Main Camera and Directional Light, is represented in this view.



11. Show students how to navigate the Scene View using the mouse and keyboard. Highlight the option of using the right mouse button to rotate and the WASD keys to move. This will be familiar to any students who have played a first person game on a PC. Explain that when they are moving the view around they are manipulating the Scene Camera.

12. Tell students that they are going to add some objects to make their scene more interesting. Direct them to right-click in the Hierarchy Window and select 3D Object>Cube. This will create a cube in the center of the Scene View

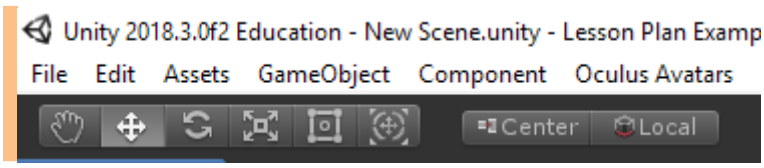


13. Explain that this is the default cube. It is one meter long on all sides and can be used to gauge how large other objects are. It may help students to relate a meter to some common objects.

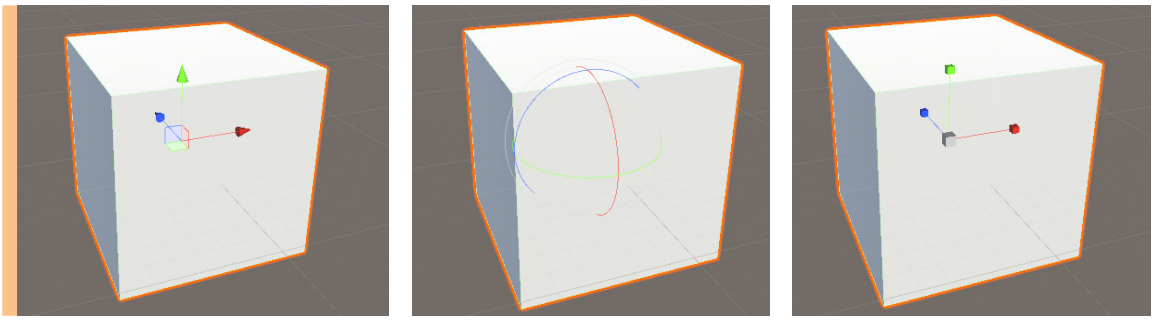
- How high is a table in meters?
- How tall is a person?
- How long is a car?

Questions and Answers

Show students how to select the position, rotation and scale tools by clicking on the buttons in the Toolbar in the top left corner of the Unity window.



Demonstrate the use of the tool handles in the Scene View. Show that clicking on one of the red, green or blue lines and dragging with the mouse will manipulate an object. Explain that clicking and dragging the center of the scale tool will scale an object uniformly.



12. Give them a small amount of time to move around the scene and practice using the controls. Have them refer to the Unity Interface Cheat Sheet if they get stuck. Walk around during this time to check for understanding. Make sure all students have completed at least the following tasks.

- Practice moving the Scene Camera using the mouse buttons.
- Use the right mouse button and the WASD keys to navigate a scene.
- Create a cube using the Create menu in the Hierarchy Window
- Change the position, rotation and scale of the cube using the Scene View tools.

Unity Activity

Tell students that the cubes they created will probably not be visible to the Main Camera, and they wouldn't be able to see them if they put on the VR headsets. Explain that every scene has a center point known as the origin and if they move a cube there, they will be able to see it. Relate the origin of the scene to the origin of a 2D graph.

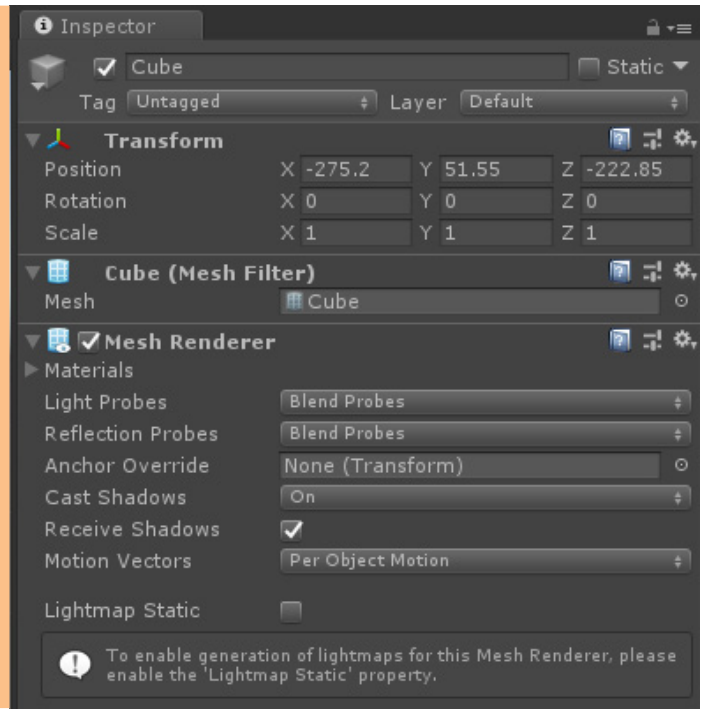
- What is the center point of a 2D graph?
- If we move right on a graph, do the numbers increase or decrease?
- What if we move up and down?

Questions and Answers

Show students how to use the Inspector Window. Focus their attention on the Transform component. Demonstrate that the values for position, rotation and scale can be changed by typing in numbers. Instruct them to type in 0 for the X, Y and Z values of the position field.

- Why would we want to type in numbers instead of using the tools in the Scene View?

Questions and Answers

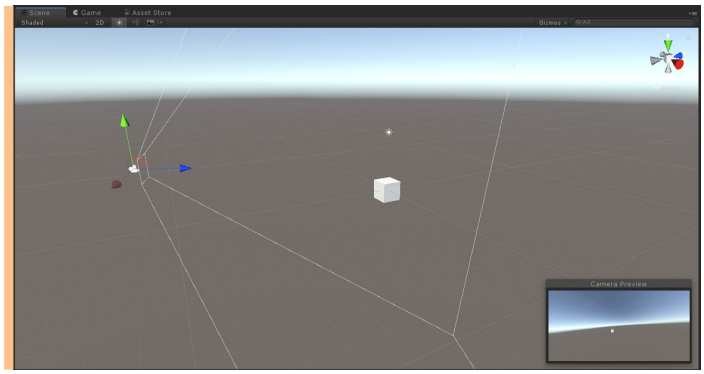


Explain that the Inspector can be used to modify the components of individual objects. Point out that similar to how the scene is a collection of objects, an object is a collection of components. Use the cube as an example and point out the Transform, Mesh Filter, Mesh Renderer and Box Collider components.

- What do you think these components do?
- Why might it be a good idea to make an object out of components?v

Questions and Answers

Explain to students that the Main Camera object will match the position and rotation of the tracked headset and will display what it sees on the headset lenses. Show that by clicking on the Main Camera in the Hierarchy, white lines will appear in the Scene View to delineate the camera's view.



Show students the Play button in the Toolbar. Explain that this button allows the program to run exactly as it would if it was a separate application. Instruct them to press the button and put on their VR headsets. Show that any changes made while in Play Mode will be undone when exiting Play Mode.

- Make sure all students' VR headsets are tracking. If they are not, make sure any required software such as Oculus Home or SteamVR is installed and running.
- If students can't see the cube, make sure they placed it close enough to the camera.
- Note that if the camera is inside the cube(because it is too close or too big), they will not be able to see it.
- Students who are ahead of the others can experiment with some of the other shapes in the Create menu.

Unity Activity

When all students have completed the tasks outlined above, gather their attention and discuss the following questions.

- What are the benefits/drawbacks of having the scene reset when exiting Play Mode?
- What is the difference between seeing something on the monitor and in VR?
- What kinds of things could we create with the cube primitives? Walls? Rooms? Buildings? More complex shapes?

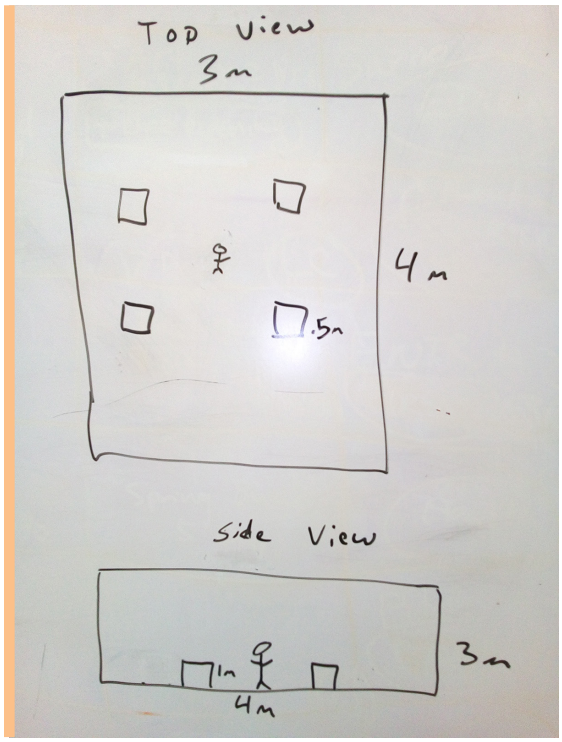
Questions and Answers

Tell sudents that you are going to give them a tool that will help them in all aspects of designing and creating a VR experience. Distribute Game Development Journals. Make sure students understand the importance of the design journal. They must bring it to class every time as it will be used in all of the lessons. If there is time, let the students personalize their journals in some way

Tell students that taking time to plan will save time later. Explain that sketching can be a very helpful tool for planning and also sharing their ideas with others. Demonstrate how to make a simple floor plan style sketch. Make sure to mark lengths in meters. Students will then make their own sketches in their journals.

- Sketch what your room will look like in a top down view. Label the sides with the length and width.
- Sketch a side view and mark the height of the walls.
- Make a mark to show where the player will be in the room.
- Draw any other objects that will be in the room and mark their size.

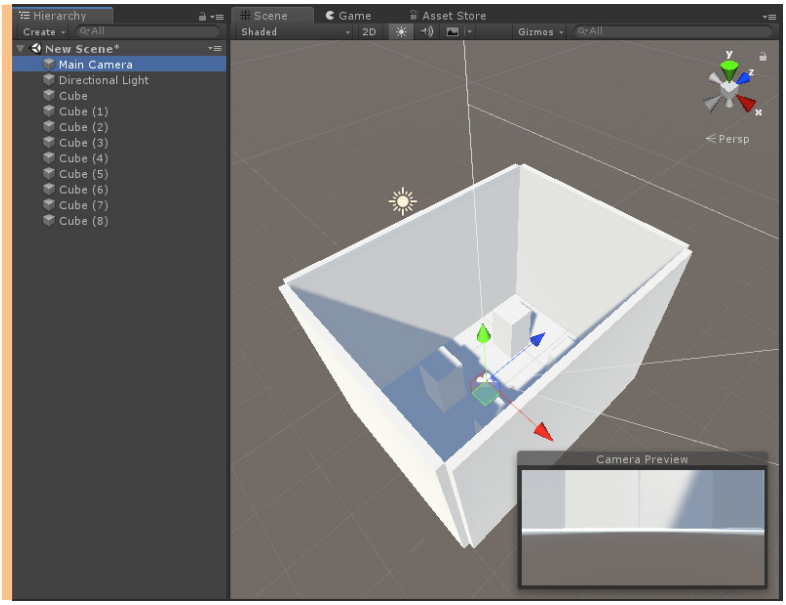
Game Development Journal



Show students how to scale a cube so that it looks like a wall. Show them how to enter exact sizes using the Inspector. Remind them that cubes are one meter on each side by default, so a cube scaled by 3 in the X direction will be 3 meters long.

Explain that they can create duplicates of items by right-clicking on the object in the Hierarchy Window and selecting Duplicate, or using the Ctrl+D shortcut. Make copies of the cube wall and modify their position and rotation to make a room.

Explain that they can control where the player will start by moving the Main Camera. Show how to use the position tool to move the camera to where we showed in the sketch. Explain that they can change the starting rotation of the player by using the rotate tool. Point out that the blue arrow of the camera is the forward direction.



Give students time to translate their sketches to Unity.

Unity Activity

- Create cubes of the desired size using the scale tool or the Inspector.
- Rotate and position them using the tools or the inspector.
- Move the main camera to the position defined in your sketch.
- Test the scene to see how it looks and modify as needed.

Teaching Tips and modifications:

This lesson covers a lot of features. If time runs out before the end of the lesson the task of making a room can be split and done in the next lesson. Some students may be well ahead of the others. They can experiment with the various primitive shapes present in the create menu, the lights or the particle systems.

Outcomes

Students should come away from this first lesson with a grasp on the basic tools. This lesson should also start to give them ideas as to the kinds of worlds they can make in Unity.

Lesson 2: Demonstration Project

Overview

This demo project will get a little deeper into the Unity environment. The goal is to keep things interesting while teaching some things that students will be able to use in their own projects.

The project linked below is a complete demo with all the needed plug-ins for the Vive and Oculus. There are also some sounds, materials, prefabs and a complete scene.

Learning Objectives	**Students will:** 1. Follow along to create a simple demo project. 2. Practice using the Unity environment. 3. Learn more advanced options in the Unity interface 4. Create a terrain 5. Use a custom script to play a sound when one object collides with another
Materials	• Unity demo project (Snowball Fight scene)
Links	https://docs.unity3d.com/Manual/Prefabs.html https://unity3d.com/learn/tutorials/topics/physics/rigidbodies
Suggested Time	2 hours
Suggested Grouping	Single Students
Vocabulary	-terrain -component -scripting -prefab -animation -parent/child

Preparation

- Download demo project to all computers.
- Make sure the VR headsets are working before starting class.
- Use the lesson plan to create your own version of the demo project as an example.

Procedure

1. The terrain system is an exciting tool for students to use and should keep them interested

b. Explain the basics of the terrain system

1. Raise/lower terrain
2. Change brush shape, size and opacity

c. Explain some of the limitations of the terrain

3. No holes/caves
4. Can't rotate or scale

2. Explain how to create a flat spot for the player/snowman by lowering the terrain

Unity Activity

- Create a terrain and make mountains
- Flatten terrain in one area for the player and snowman

3. Set the player down

a. Use the VR Setup prefab in the Prefabs folder

b. Drag the prefab into the scene where the player will be

Unity Activity

- Place the VR setup prefab as needed
- Check to make sure your controllers are visible in play mode–

4. Create the Snowman

a. Students should already know how to create and scale objects

b. Demonstrate how to make objects children of other objects

c. Explain why parent/child relationships are important to work flow

Unity Activity

- Make three spheres. One big, one medium and one small
- Stack the spheres
- Make two of the spheres a child of the third
- Add decoration, e.g. eyes, nose, hat, etc.
- Check to make sure it is visible and properly scaled in play mode

4. Drag snowball prefabs into the scene

6. Demonstrate the use of Prefabs

a. Explain what a Prefab is and why it is useful

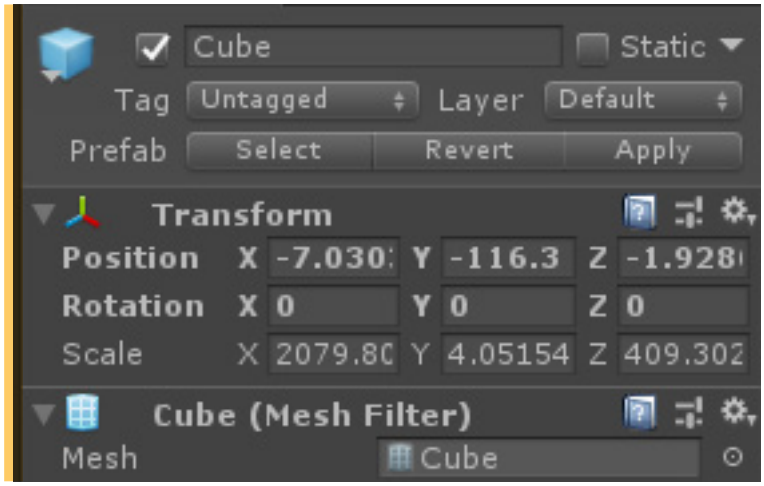
1. They make it easy to edit multiple objects
2. They can be reused across multiple scenes

b. Make a prefab of the snowball object

1. Drag it from the Hierarchy into a folder in the Project Window
2. Creating a new folder just for Prefabs is recommended

c. Drag multiple copies of the snowball prefab into the scene and show how to edit them all at once

1. Edit some component of one, e.g. the scale property
2. Apply the changes to all instances of the snowball by clicking the



7. When the snowballs hit the snowman, play a sound

a. Find the DoWhenCollisionEntered script in the Scripts folder

b. Drag it onto the top level of the snowman(the one that is a parent to the others

c. Add a Rigidbody component to the same object

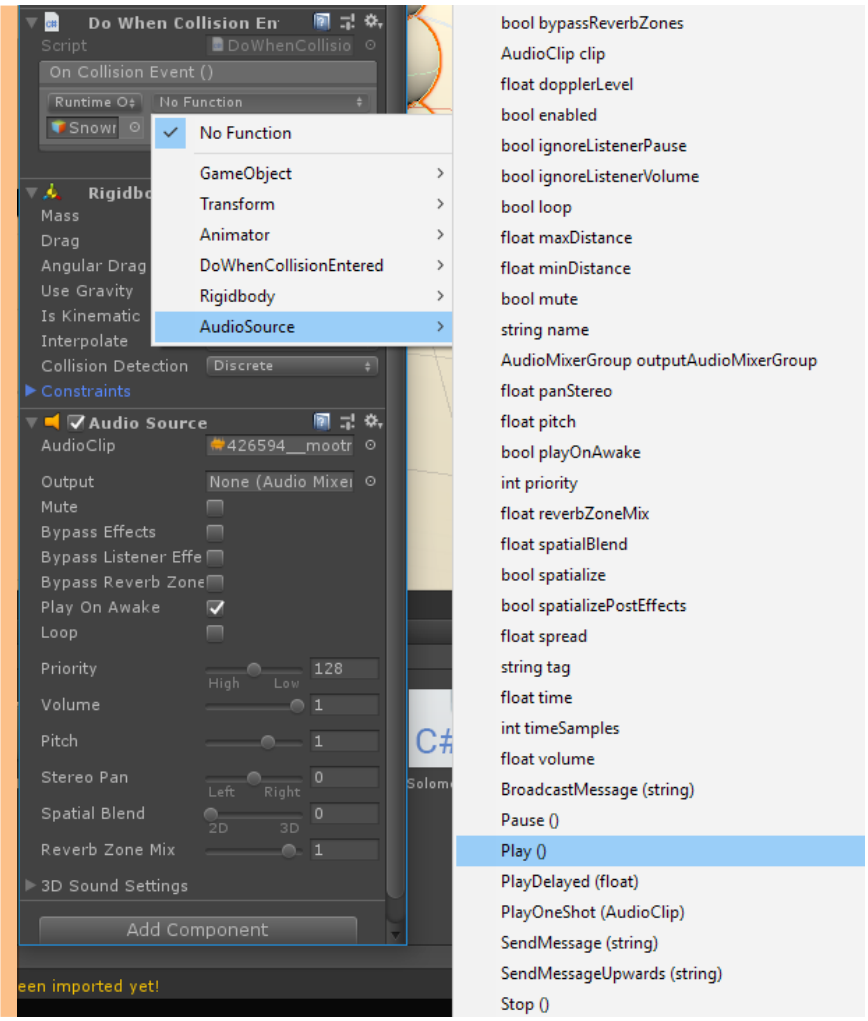
d. Drag a sound file from the Sounds folder onto the same object

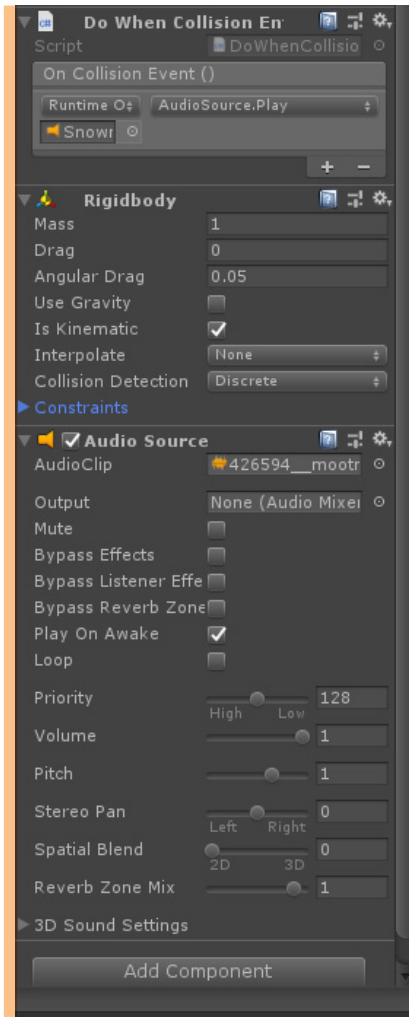
e. Press the + sign at the bottom of the DoWhenCollisionEntered component

f. Drag the Snowman object from the Hierarchy Window into the field that was created

g. Click the button that says No Function and navigate to the AudioSource > Play function

I. On the Rigidbody component, check the box marked Is Kinematic. This will prevent the snowman from falling over.





K. The final setup should look like this

8. This is a good point to stop and let students catch up and give help where needed. Some common problems will crop up, especially during the last step.

a. The collision script will only work if the parent object has a Rigidbody component attached

1. If the Rigidbody is not set to kinematic, the object will respond to gravity and other physical forces, which may be undesirable

b. The AudioSource component must have a file in the AudioClip property

c. The DoWhenCollisionEntered script must have the correct AudioSource and the correct function in the action field.

9. Once the students have all of these objects working, ask the following question

- What other kinds of things do you think Unity can do?
- What do the components do? What others are available to us?

Q/A

10. The students should use their design journals to brainstorm what improvements could be made to the demo project they just completed

- What could be done to improve upon the snowball game in any respect(art, features, gameplay etc)?
- What components would be used for these improvements?

Game Design Journal

Outcomes

Students should be getting more comfortable with the editor and start to see some of the power behind it. They should be able to utilize some of the most common features, such as parent/child relationships, prefabs and adding components.

Teaching Tips and modifications:

Depending on how many students you have and how fast they are working, you can explore some of the following topics.

1. Particle Systems(Snow, hit effects, etc)
2. Terrain painting
3. Writing the DoWhenCollisionEntered script from scratch
4. Animating the Snowman to make a more difficult target to hit
5. Changing the properties of the Rigidbody on the snowball object. High angular drag will stop it from rolling as much.
6. Creating a platform to hold the snowballs to make it easier to pick them up
7. Adding more actions to take using the DoWhenCollisionEntered script. It can play Particle Systems, turn on/off other objects and more
8. Adding the DoWhenCollisionEntered script to the snowballs to play sounds when they hit something

Lesson 3: An introduction to VRTK
A cross-platform toolkit

Overview

VRTK (VIrtual Reality Tool Kit) is a collection of useful scripts and prefabs for building VR titles in Unity3D. It offers common functionality for major VR platforms including the Oculus Rift and HTC Vive. This lesson will cover the basic setup of VRTK and explain the purpose of some of the components.

Learning Objectives	**Students will:** 1. Define the concept of an SDK 2. Identify the SDKs used in this lesson 3. Define the purpose of VRTK 4. Outline the critical components of VRTK 5. Apply the components listed above to a blank scene
Materials	Unity Demo Project
Suggested Time	2 hours
Suggested Grouping	Pairs

Links	• https://vrtoolkit.readme.io/docs/summary
Vocabulary	• -VRTK -

Preparation

Procedure

1. **Warm-Up:**

   a. E

2. Discuss the following concepts as they apply to a game

a. Th

b. Start a discussion about a different popular game and try to fill in the four main components

- Pick a well known game.
- What is the story/setting of the game?
- What are the goals, obstacles and abilities?

Questions and Answers

3. Show students examples of completed GDDs. Point out the key features in the design document

a. Story/setting, goals, obstacles, abilities

5. Create the snowball and make it interactable

a. Make a new sphere and demonstrate how to rename an object

b. VRTK offers an easy way to make objects that can be interacted with by the controllers

c. Go to Window > VRTK > Setup Interactable Object

d. Explain the important options. Demonstrate how to make objects children of other objects

1. Highlight Color
2. Hold Button to Grab

Unity Activity

- Create a new sphere
- Make it interactable
- Test in play mode

- Sketch a layout of a scene in the game.
- Define where the player can and cannot go.
- Show what obstacles the player will overcome.

Game Development Journal

Teaching Tips and modifications:

There are many experiences that forgo some or all of the above categories. A type of experience termed "walking simulator" (e.g. Dear Esther) does away almost entirely with the goals, obstacles and abilities. Others types may do away with the story, goals and obstacles and focus on creative play e.g. Minecraft. Students may want to work on something that focuses on one aspect, but caution them that it may be more difficult and less rewarding. A project that spreads itself over the four components can be engaging even if some parts are less than optimum.

Lesson 4: Introduction to Scripting

How to create custom features for our game.

Overview

Scripting in Unity allows us the freedom to customize our game and create brand new features. It is the only way to craft an interactive experience, as opposed to just a passive one. By focusing on a small subset of the features available, we can reduce the complexity of the process to a manageable level.

Learning Objectives	**Students will:** 1. Learn the purpose of scripting in game development. 2. Understand some features of the Unity API. 3. Write pseudocode to help define what features are needed. 4. Understand the basic rules of the C# scripting language.
Materials	• C# for Unity Cheat Sheet
Links	**Video tutorials for scripting. Good for all skill levels** https://unity3d.com/learn/tutorials/topics/scripting **A text resource to learn the how scripts work in Unity.** https://docs.unity3d.com/Manual/ScriptingSection.html **Quick reference for common Unity scripting features** https://www.raywenderlich.com/227-unity-cheat-sheet-and-quick-reference-2018
Suggested Time	2 hours
Suggested Grouping	Single Students
Vocabulary	-scripting -syntax -API(Application Programming Interface) -logic -C# (C sharp) -pseudocode -component

Preparation

- Make copies of the C# for Unity cheat sheet for each student.
- Teachers should review the lesson and have a good grasp on the use of scripting in Unity.
- The linked videos help teachers learn how scripting works in Unity.
- Make sure Microsoft Visual Studio is installed on the computers. It requires a free account with Microsoft. One account can be used for all installations.

Procedure

a. Address the following questions with students.

- What do they think scripting(or programming) is?
- What kind of things can we do with scripting?
- How are scripts similar to the built in components, such as the Animator, Rigidbody, etc?

Questions and Answers

2. Discuss the kinds of things that can be done with scripts. Explain that any setting that can be changed in the Inspector can also be changed via a script. Demonstrate some of the items that can be changed in the Inspector.

a. Change an object's color.

b. Move an object to a new position

c. Change the settings of a Rigidbody

3. Scripts also allow access to functionality that is not available in the Inspector.

a. Add forces to a Rigidbody

b. Play a sound when objects collide

c. Keep track of data like score and player health.

4. Explain that custom code can be executed at certain points.

a. The Start function runs code at the start of the game

b. The Update function runs code once per frame

c. The OnTrigger/OnCollider functions run code when objects collide

5. Discuss the concept of pseudocode. Explain that it is like sketching for programmers.

- Why would we want to write pseudocode? Why not just go straight to the final code?

Questions and Answers

5. Show the students how to write pseudocode for the function described later in this lesson. This pseudocode is a rough outline of what we want to happen when an object collides.

a. Make a new random color.

b. **Go to the renderer of this object.**

c. Go to the material on the renderer and change the color property.

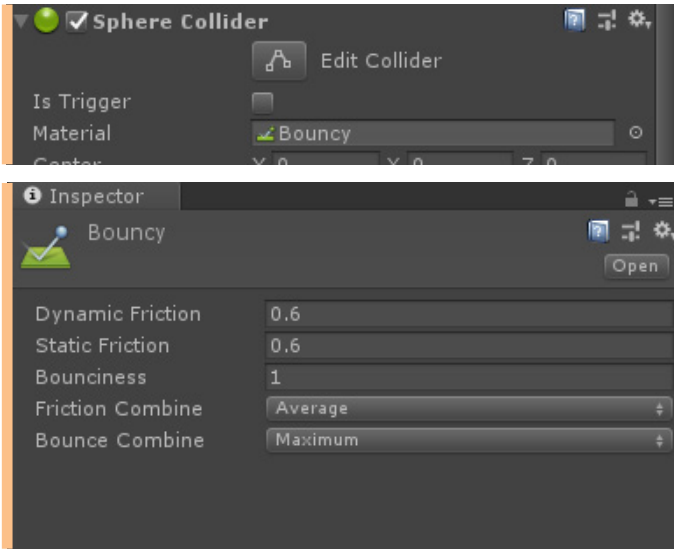
5. Explain that now you will write the real code to make the color change happen. Start by creating a Sphere.

a. Add a Rigidbody component to it to allow it to fall and bounce.

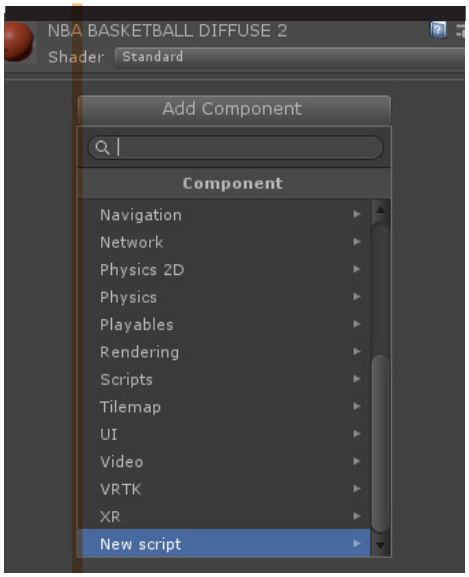
b. Make a bouncy physic material by going to Assets>Create>Physic Material.

c. Click the newly created material in the Project Window, then change the Bounciness property to 1.

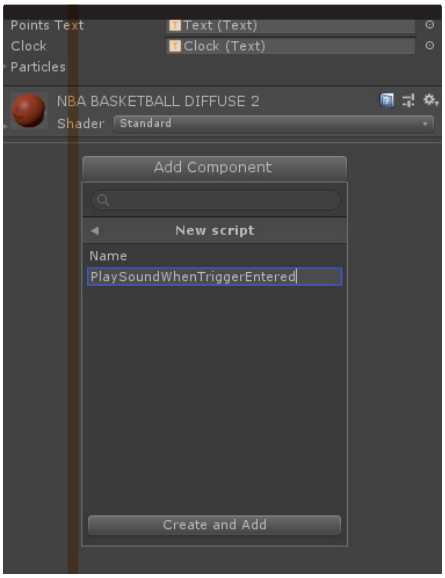
d. Drag this to the Material slot on the Sphere Collider component on the Sphere.



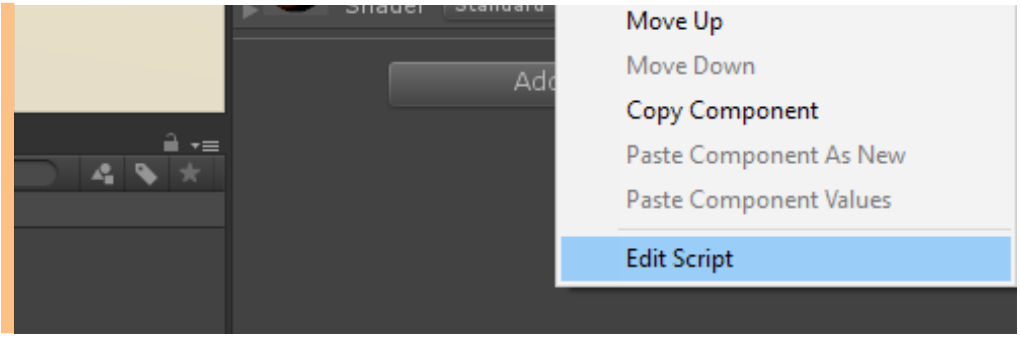
9. Click the Add Component button and scroll to the bottom. Click New Script.



10. Type a name for your script in the text box



11. Click the gear icon and then click the edit script button. This will open your default script editor.



12. When Visual Studio opens, you will see this default template. The Start and Update functions can be deleted.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class ChangeColorWhenCollidedWith : MonoBehaviour
6   {
7
8       // Use this for initialization
9       void Start () {
10
11      }
12
```

Show students how to write the final version of this code. Use the following steps.

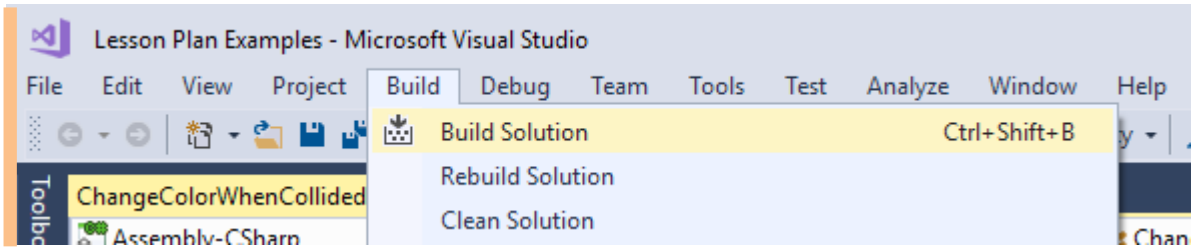
- Get a component attached to this game object. The type of component we want to get is determined by what is between the angle brackets. In this case <Renderer>
- Go to the material property on the renderer.
- Go to the color property on the material.
- Set the color to the color that we declared above.
- Get a component attached to the game object. The type of component we want to get is determined by what is between the angle brackets. In this case <Renderer>
- Go to the material property on the renderer.
- Go to the color property on the material.
- Set the color to the random color that we declared above.

Give them a copy of the C# Cheat Sheet so they can follow the syntax. This can be found in the Appendices section.

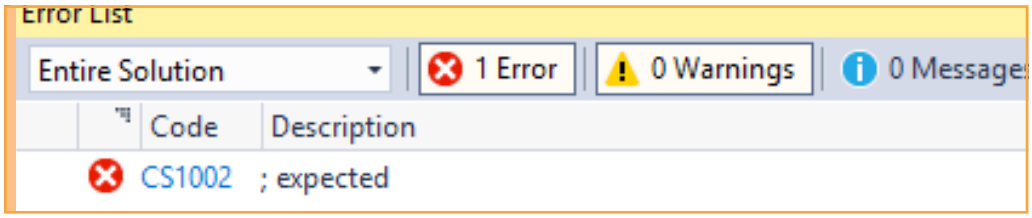
14. Your final code should look like this. Give students time to digest the code and prompt them for questions.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ChangeColorWhenCollidedWith : MonoBehaviour {

    private void OnCollisionEnter(Collision collision)
    {
        Color randomColor = new Color(Random.Range(0,1f), Random.Range(0,1f), Random.Range(0,1f));

        gameObject.GetComponent<Renderer>().material.color = randomColor;
    }
}
```

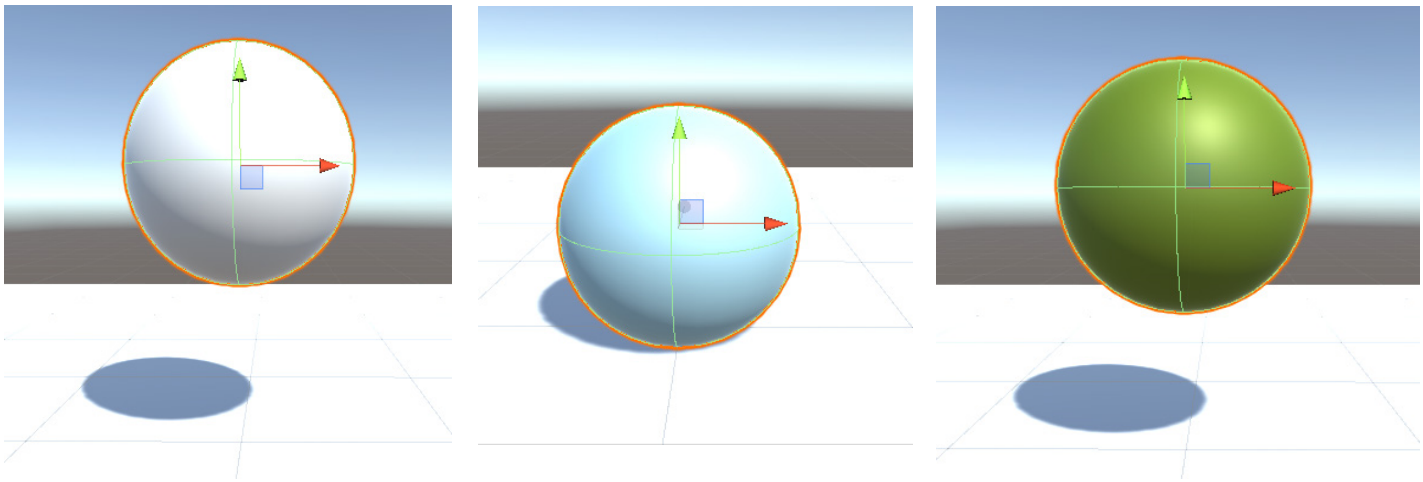
19. Show how to compile the script (also known as Build). Explain that this step will check for errors and make the code readable to the machine.



19. If your code has any errors they will be displayed in the box at the bottom of the window.



19. When your code is error free, go back to Unity and play the scene. The sphere will bounce on the plane and change color every time it hits.



20. Direct the students to create a script of their own. Have them start by looking in the Inspector for a property to modify.

- Write pseudocode outlining the functionality you want to achieve

Game Development Journal

20. As students are writing their pseudocode, walk around and give directions and help as needed. When students have finished writing pseudocode in their journals they can start implementing it in Unity.

Unity Activity

- Create a new script from the editor
- Implement one of the Unity Messages
- Modify a property of some GameObject using statements inside the message

20. Walk around and help students write the final code. Have them refer to the C# Cheat Sheet for the basic syntax and common errors. Have them test their code when they are finished

Teaching Tips and modifications:

There will be many questions about why things are the way they are. If you have a deeper knowledge of the language or computer science, feel free to pass it on to the students but try to keep things focused on the concrete examples. It is OK to say, "Just because" at a certain point.

Encourage the students to look at the links provided for more information. If needed, this lesson can be split into multiple sessions. Plan to have a spare day dedicated to scripting. If you manage to cover everything in a single lesson, the second day can be dedicated to work time.

Lesson 6: 3D Modeling in VR

Create 3D Models with VR tools

Overview

The Unity primitives(cubes, spheres, etc) allow us to create a rough 3D sketch of our experience and are a critical stepping stone for our project. However, we will want finer control over the look of the objects in our scene. We get this control in an external 3D modeling package.

A VR based modeling program is a fun and intuitive way to explore the basic concepts of modeling in a simplified environment.

Learning Objectives	**Students will:** 1. Understand the concept of a background, middleground and foreground. 2. Understand basic 3D modeling terminology and practices that apply to any modeling tool. 3. Learn the strengths and weaknesses of a VR modeling tool 4. Create a sketch of an object they want to model 5. Learn the basics of working with a VR modeling program. 6. Import the finished model into their Unity projects
Materials	• Modeling cheat sheet • Copies of Medium for as many computers as possible
Suggested Time	• 2 hours
Suggested Grouping	• Single Students

Links	**A lesson plan exploring fore, mid and background as it applies to paintings** • http://www.getty.edu/education/teachers/classroom_resources/curricula/landscapes/lesson02.html **A collection of videos from the Oculus Medium team that shows how to use the software** • https://www.oculus.com/medium/learn/
Vocabulary	-polygon -texture -polycount -mesh -geometry -3D modeling tool

Preparation

- Make copies of the cheat sheet and the fore/mid/background worksheet. Have working copies of Medium on one or more computers.

Procedure

1. Warm-Up:

- Why do we need to create 3D models?
- How can our projects be improved by the addition of custom 3D models?
- What kind of things could we create to improve our game?

Questions and Answers

2. Discuss the concept of a background, middleground and foreground.

a. Background is the farthest away from the player. It is typically a flat image that every other object is drawn in front of

b. It is usually called a skybox because it is 6 images that are placed on the interior of a cube

c. The middleground contains objects that may be 3D models, but often are so far from the player that they are not able to be seen as different images from each eye(stereoscopic vision)

d. The foreground contains objects that are close enough for the player to see them as 3D objects and may be things the player interacts with.

- **Ask the following questions**

- **Where do we need the most 3D detail? In the fore, middle, or background. For an object very far from the player how much detail do we want to give it?**

Questions and Answers

3. Before students get into sculpting with Medium, have them sketch a design of what they want to create in their design journals

4. Do a demonstration of how to use Medium. This demo should be no more than 10 minutes total and is just to help orient the students. Be sure to cover the following points.

a. The working hand and the support hand and the menus available on each.

b. The clay tool.

1. Adding and subtracting clay
2. Using the different primitives (sphere, cube, capsule) to create rough shapes
3. Using the extended shapes(stamps) to add finer detail.

c. The Undo/Redo command on the support hand thumbstick.

d. Using the move tool to pull clay into the desired shape.

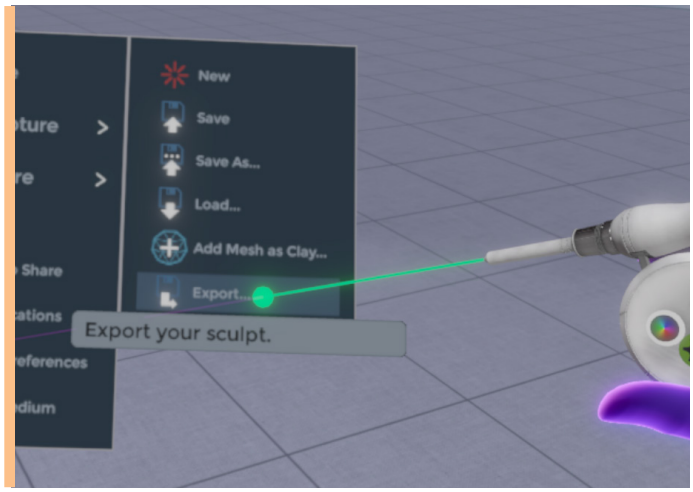
e. Using the smooth tool to clean up areas and blend shapes together

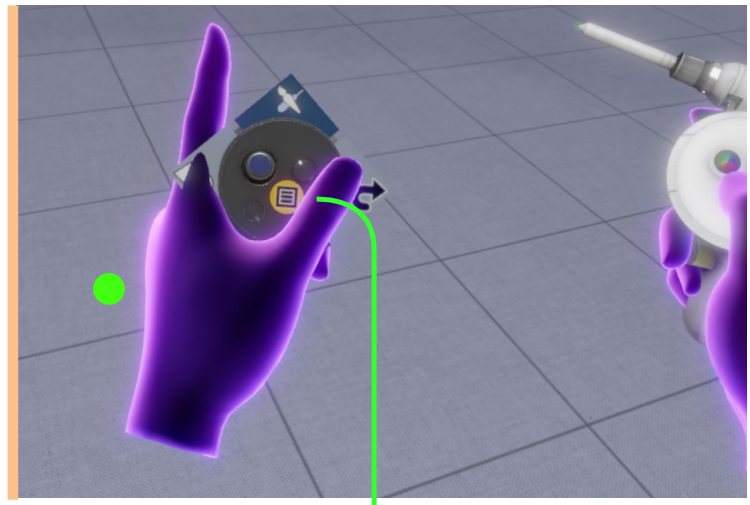
5. Allow students to build their sketches in Medium. As they are working give individual advice.

6. After they have had some time to create models, show them how to export from Medium.

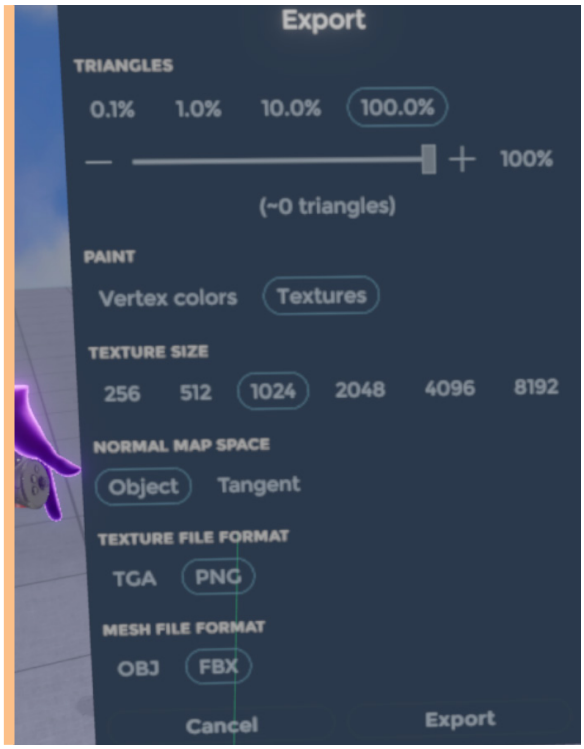
7. The export menu is found by clicking the yellow button on the support hand

8. Click the File button and select Export under the drop down





Click the File button and select Export under the drop down

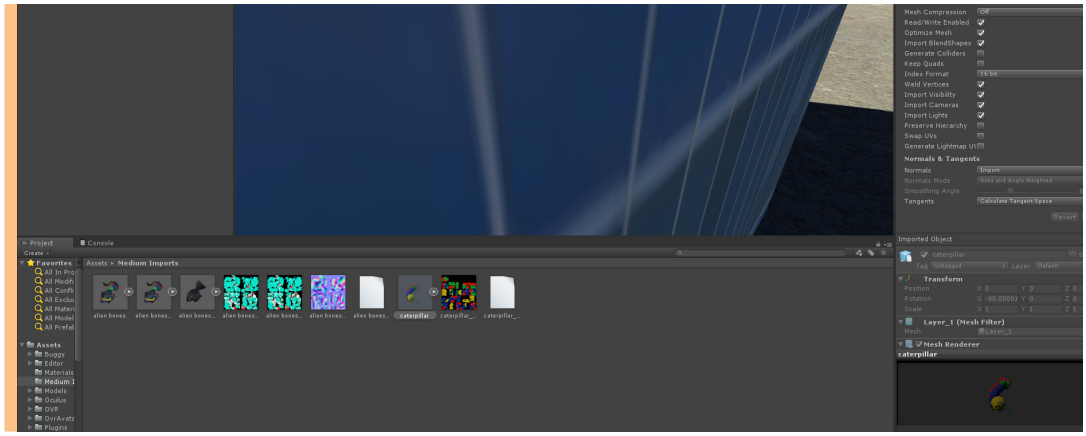


8. continued

1. The slider at the top determines the final poly-count(counted as Triangles) of the model.
2. Students should have less than 200k polygons for a very detailed foreground model, many less for background models.
3. Texture colors (not vertex) should be selected. Size for textures should be no more than 2048.
4. Tangent or normal space doesn't matter.
5. PNG is best for Unity.
6. FBX is best for Unity.
7. Models are placed in the Documents/Medium/Exports/ folder by default.
8. Exporting happens even if the headset is removed

9. Discuss the strengths of a VR modeling tool

10. Drag Models into the Project Window



11. Drag the object from the Project Window into the Scene View

12. Objects may show just gray color. To give it the colors that it had in Medium, drag the texture file onto the model in the Scene View

- Open the folder where the model was saved.
- Drag and drop all the files into the Project Window in Unity.
- Make sure textures are dragged in as well.
- Once the asset is in the project, it can be dragged into the Scene View.
- The texture can be applied by dragging it onto the model in the scene.
- The student can manipulate the object texture like other objects.

Unity Activity

Conclusion

- After this lesson, students should know the steps needed to create a custom 3D model and bring it into their Unity project. They should take time after the lesson to practice their skills, starting with a sketch and ending by importing a model into Unity. Here are some extra topics to discuss to give an idea of the wider world of 3D modeling.

For Discussion

1. **Discuss the basic terminology of 3D modeling**
   a. Give students the 3D modeling cheat sheet from the Appendix.
   b. Discuss the terms there and answer questions.

2. **Discuss the strengths of a VR modeling tool**
   a. Rapid sketching of ideas.
   b. Can create highly detailed, organic models, like trees, faces.
   c. Easy to learn.
   d. Easy to get the scale of an object correct (since we can see it with stereoscopic vision)

3. **Discuss the weaknesses of a VR tool**
   a. Not very accurate.
   b. Polycount can be unacceptably high.
   c. Reducing polycount with decimation can cause ugly results.
   d. Models can be more difficult to animate.
   e. Not as well supported as traditional packages.

4. **Give some examples of other 3D modeling tools**
   a. Maya, 3DS Max
   b. SketchUp, Revit, AutoCAD

- Make a list of what 3D models you will need for your scene
- List where they will be in your scene; foreground, background, or middleground

Game Development Journal

Teaching Tips and modifications:

This lesson allows everyone to get creative and is an excellent way to engage students who are more artistic. Be sure to allow lots of time to explore Medium.

Final Lesson: Presentation and Extended Goals

Show off the completed work and make plans for extending it.

Overview

The final showcase is an opportunity for students to show their work to family and friends. This can be done in an open house style showing where the students can show and explain their work.

Students build a standalone copy of their project that can run without the Unity editor and take a moment to reflect on the process.

Learning Objectives	**Students will:** 1. Create a build of their project that can be played on any VR computer. 2. Reflect on their work throughout the course. 3. Present their final project to the public.
Materials	
Links	**Overview of Building and Publishing** • https://docs.unity3d.com/Manual/PublishingBuilds.html
Suggested Time	2 hours
Suggested Grouping	Single students or Pairs
Vocabulary	-build -publish

Preparation

- Review the steps for building a project.
- Build a test VR project and make sure it runs properly. There is a chance that bugs show up in the build version but not in the editor.

Procedure

1. Tell students that they will build their projects into standalone programs that do not require the Unity editor to run. Explain that this what commercial game developers do before they release their product. Point out that this will strip out any unused assets from the final project, thus reducing the size.

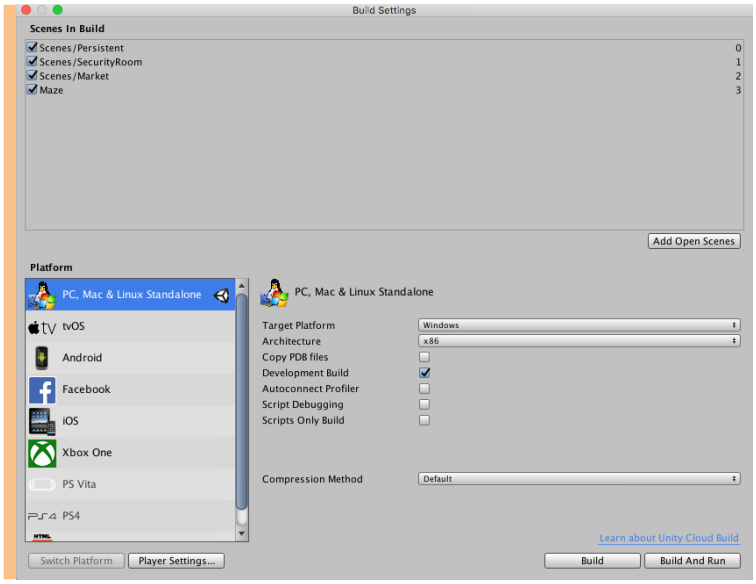
2. Demonstrate the work flow for building a project

d. Open the scene that you want to include in the built project.

f. Go to File > Build Settings

g. Click the Add Open Scenes button

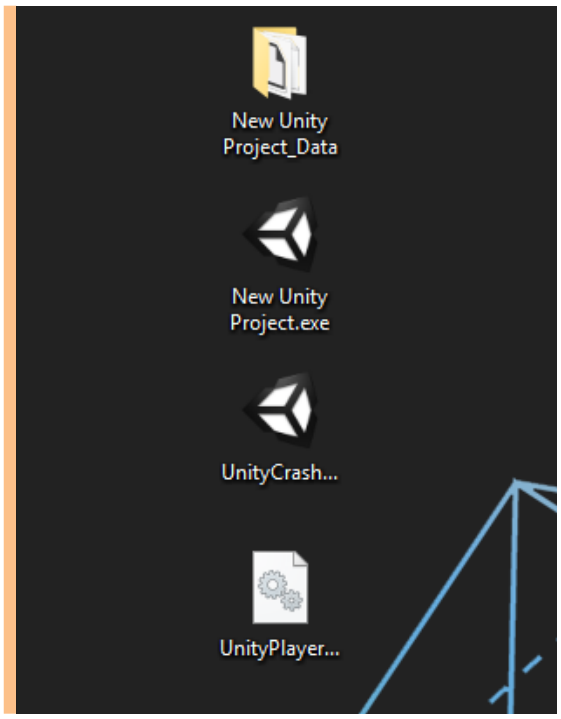
h. Make sure the Platform is set to Standalone.



i. Click the Build button.

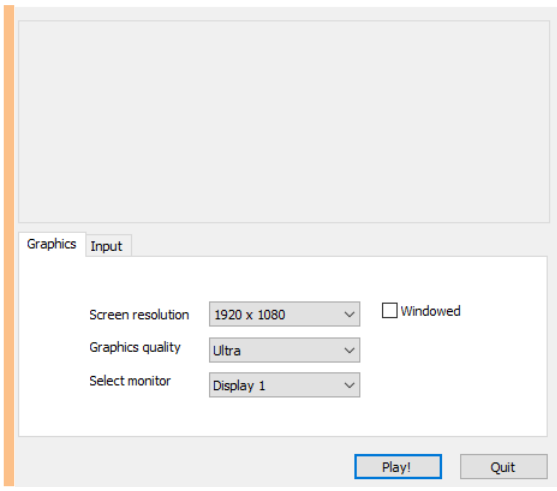
j. Select where the built project should be saved. This should not be inside the assets folder of the Unity project.

k. The build process will commence. Depending on the complexity of the project, this can take from minutes to hours.



l. When the process is complete, there will be several new files in the location you specified.

m. All four of these files must be present in the same directory for the executable to work. If a student wants to take the built project home, they MUST have all of these files.



n. To play the project, open the .exe file with the project name on it. You will be presented with a configuration dialog.

o. Click Play! and the project will open, just as if it was playing in the editor.

p. Test all functionality, since things can change between the editor and the build.

3. Give students time to build their own projects

- Students should build and test their projects.

Unity Activity

3. Once projects are working, take a few minutes and talk to students about their experience.

- What did they enjoy about creating a VR project.
- If they did the project again, what would they change?
- What projects would they like to create in the future?

Questions and Answers

4. This is a good opportunity for students to give feedback on the course. If you have a post course survey, have them fill it out before the public showcase.

5. The public showcase should be about an hour for parents, friends and others to explore the projects created by the students. It may be enlightening for parents to try some of the other VR titles and tools that the students used during the course. Make sure to leave time for closing remarks and thank yous to parents and students.

Teaching Tips and modifications:

The open house can be whatever format works best for your location. Opening it to the public can help drive interest for further programs.

Another option for showcasing student work is to publish it on a website. Itch.io is a great website that allows you to upload your projects for the world to see and download. Some students might be interested in creating the art for the pages on itch.io.

Unity Interface Cheat Sheet

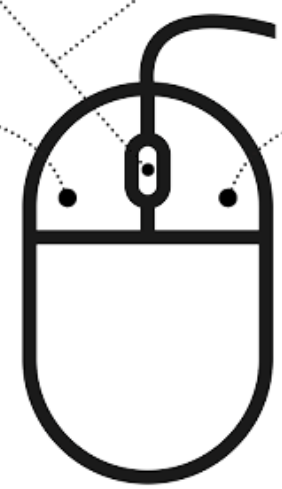
Keyboard/mouse controls and common terms

Use SCROLL to zoom the Scene Camera in and out.

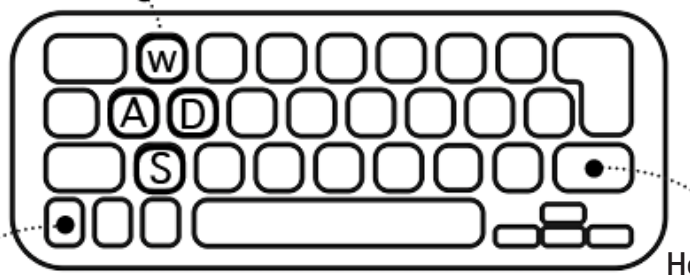
Hold MIDDLE MOUSE and move mouse to pan the Scene Camera left, right, up and down.

Click LEFT MOUSE to select an object in the Hierarchy Window or Scene View.

Hold RIGHT MOUSE and move the mouse to rotate the Scene Camera. Also used for WASD movement.

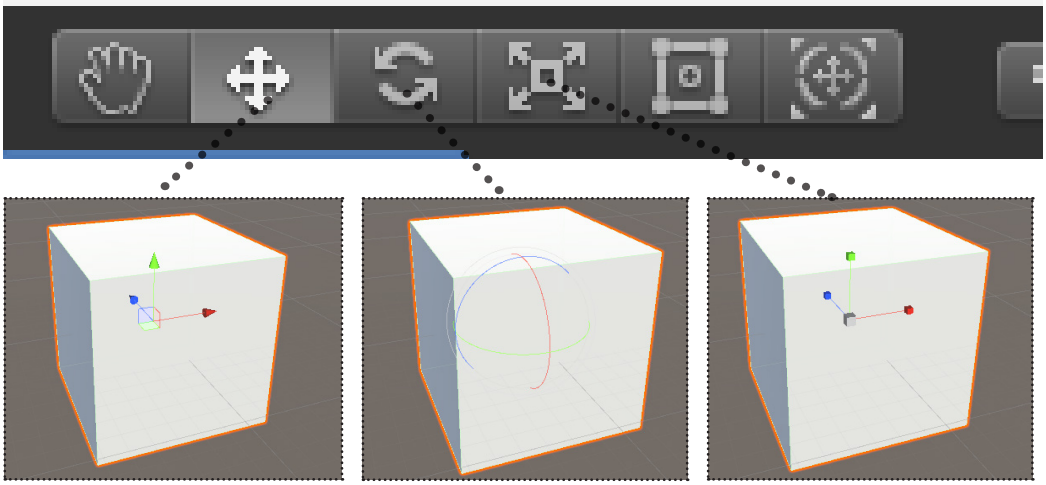


While holding the RIGHT MOUSE button use WASD to move the Scene Camera forward, backward, left and right.



Hold CTRL while left-clicking to select multiple objects in the Hierarchy or Scene View.

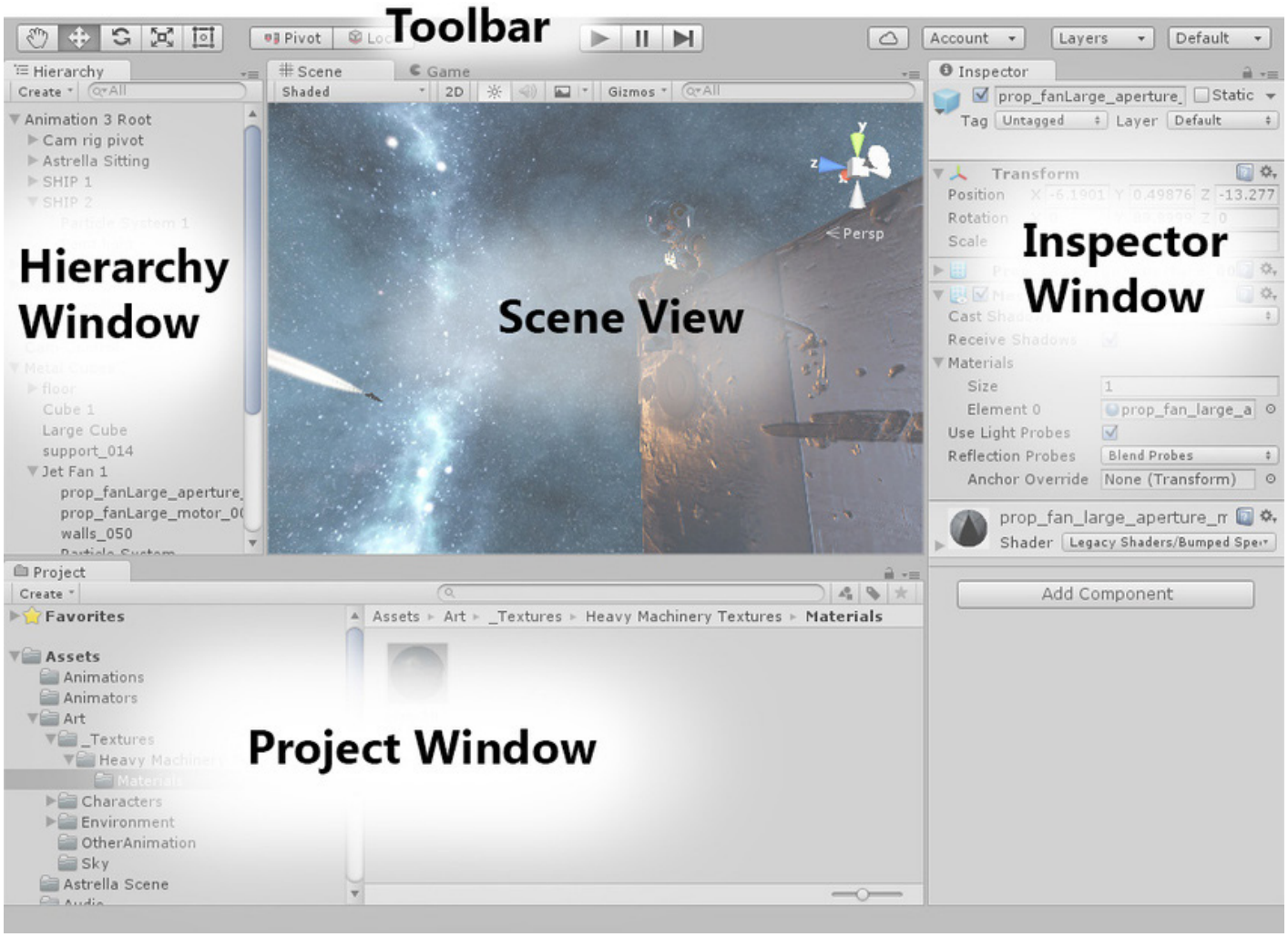
Hold SHIFT while left-clicking to select a range of objects in the Hierarchy Window. Holding SHIFT will also speed up movement when using WASD keys.



Position Tool:
Click on colored arrows or the center and drag to move an object.

Rotate Tool: Click on the colored lines and drag to rotate an object.

Scale Tool: Click on the colored cubes or the center cube and drag to change an object's size.



C# for Unity Cheat Sheet

Basic syntax and common Unity API functions

The semicolon is like a period. It shows the end of a statement. Most lines in C# will end with a semicolon.	transform.Position=new Vector3(3,6,9);
	playerScore=playerScore+1;
	objectMaterial.color=Color.blue;

Curly brackets are used to contain a block of code. They always come in pairs. So a { is always paired with a }.	void WinGame()
	{
	displayText.text="You Win!";
	}

If statements let our code take different paths.	if(playerLives < 0)
	{
	displayText.text="Game Over";
	}

Parentheses surround inputs to functions and if statements. Like brackets, they always come in pairs, so a (is always paired with a).	Destroy(enemyObject);
Variables are like containers for data. They let us store various types of information and must be declared before they can be used.	int playerLives = 3; Vector3 checkpointPosition; string playerName="Bob";

- Unity allows us to run our code at certain points. The full list of where our code can run can be found under the Messages section at the following link https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

Start: Runs our code when the game first starts as long as the script is enabled with the check box next to the script in the Inspector.

```
void Start()
{
    playerStartPosition=transform.Position();
}
```

Update: Runs our code every frame while the game is running and the script is enabled with the check box next to the script in the Inspector.

```
void Update()
{
    if(Input.GetButtonDown("Escape"))
        {
            Application.Quit();
        }
```

OnCollisionEnter: This runs our code once when the object this script is attached to collides with something. The object must have a collider and a Rigidbody attached.

```
void OnCollisionEnter(Collision collision)
{
    if (collision.relativeVelocity.magnitude > 2)
    {
        soundEffect.Play();
    }
}
```

OnTriggerEnter: This is similar to the message above, but only runs our code if the collider attached is marked as a trigger. One of the objects involved in the collision needs to have a Rigidbody attached.

```
void OnTriggerEnter(Collider other)
{
    Destroy(other.gameObject);
}
```

- Unity has a few shortcuts for commonly used components.

Every GameObject in the scene can be manipulated from a script attached to the object. We can do things like enable/disable the entire object, destroy it, or find specific components on it or its children. You can access the GameObject that this script is attached to by typing gameObject with a lowercase .

```
void OnCollisionEnter (Collision other) {
    //Destroy the GameObject that this script is attached to. This
will also destroy this script.
    Destroy(gameObject);
}
```

Every GameObject has a Transform component that we can use to adjust the position, rotation and scale of an object. When we want to access this component we can type transform with a lowercase t.

```
void Start () {
    //Reset the position, rotation and scale of this object to default values;
    transform.position = Vector3.zero;
    transform.rotation = Quaternion.identity;
    transform.localScale = Vector3.one;
}
```

- The lowercase letters are important for both of the above examples. Using an uppercase letter will result in compile errors.
- Common syntax errors and the resulting error messages. Compile errors will show what line the error occurred on. Double clicking the error message will take you to that line.

Missing semicolon. Semicolons are used at the end of almost every line	Error CS1002 ; expected
Missing brackets or parentheses. All brackets and parentheses must come in pairs.	Error CS1513 } expected Error CS1026) expected
Class name and file name mismatch. The name of the file must be the same as the name on the line that declares the class. This one does not show up as an error in the compiler and only shows up in the Inspector in Unity.	The associated script can not be loaded. Please fix any compile errors and assign a valid script.
Forgot to declare a variable before using:	Error CS0103 The name 'someVariableThatIforgotToDeclare' does not exist in the current context

- Used = instead of == in an if statement. The single equals sign means "assign a value to a variable" but the double equals sign means, "check to see if these two things are the same". Depending on what you are trying to do, this may or may not give you a compile error. Always be sure to check your if statements when you are having trouble.

Using an uppercase G when trying to access the GameObject this script is attached to or a uppercase T when trying to access the Transform component.	Error CS0119 'GameObject' is a type, which is not valid in the given context Error CS0120 An object reference is required for the non-static field, method, or property 'GameObject.activeSelf'

Glossary of Vocabulary terms

3D Modeling terms

3D model: A digital object that is able to be viewed from any angle and is typically made up of many connected 2D polygons

API(Application Programming Interface): A set of data and functions that can only be accessed by writing custom scripts

Asset: A file that is within the project including but not limited to: 3D models, sound files and scripts

Background: This encompasses objects that are very far from the player and require very little fine detail.

Build: A Unity project that is packaged and able to be run without needing the editor

C# (C sharp): A programming language used by Unity to create scripts

Component: In Unity, these are pieces of code that contain a section of functionality. Many components make up an object in a game.

Edit Mode: The mode in Unity that allows the developer to make permanent changes to a scene. The counterpart is Play Mode

Foreground: This is area in a scene that the player is going to look at up close and potentially interact with and requires much fine detail.

Game Design Document: A document that outlines the design and functionality of a game.

Game Development Journal: A personal sketchbook or notebook that is used to keep track of ideas. This is a critical piece of the course.

Geometry: In the context of a 3D modeling, the vertices and edges that make up a model.

Goals: In the context of a game, these are what the player will be attempting to accomplish.

Hierarchy Window: The view in Unity that lists all of the objects in the scene.

Inspector Window: One of the main views in Unity. It shows the properties of any object or asset selected in the Hierarchy or Project Windows.

Logic: The flow of a script that determines what actions are taken and when. Proper syntax combined with correct logic creates a script that does what the developer wants.

Mesh: Another name for a 3D model

Middleground: This is an area of a scene between the foreground and background that requires some fine detail.

MVP(Minimum Viable Product): The idea that there is a small set of features that are critical to any product and must be completed before the product can be tested.

Obstacles: In the context of a game, these are what stop the player from reaching the goal

Play Mode: The mode in Unity that allows the developer to test a project without having to build a complete version. Changes made in Play Mode are undone when switching back to Edit Mode

Texture: Flat images that are wrapped around a model and show color and details of the surface.

Polycount: The number of polygons that make up a model. Using a model with fewer pol

Polygon: A 2D shape that has a certain number of sides. When many of these shapes are put together, they create a 3D model.

Primitives: Simple shapes that are available in Unity or 3D modeling tools like a cube or sphere

Glossary of Vocabulary terms

3D Modeling terms

Project: A collection of scenes and their associated assets

Project Window: The view in unity that shows all of the assets available to use in any scene.

Pseudocode: Notes that show the logic of a script without using the proper syntax. An excellent first step when creating custom scripts.

Publish: To make a complete project available for the public to use.

Build: A Unity project that is packaged and able to be run without needing the editor

Scene: A collection of assets that arranged to create one part of a game. This can be thought of as a level, or a stage

Script: A document that contains code to add custom features to Unity projects

SDK (Software Development Kit): A collection of pre-built tools that speed up the process of creating software projects

Setting: The time and place in which the story takes place.

Skybox: Flat images that are drawn very far from the player. These often show the sky and horizon.

Story: In the context of a game, this is the narrative that gives context to the player's actions and the setting

Syntax: The rules for how to write code in a particular programming language

Texture: Flat images that are wrapped around a model and show color and details of the surface.

Unity3D: A commercial game engine that can be used to create a wide variety of software

Virtual Reality: Digital experiences that replace real world visual and auditory sensory inputs with artificially created ones.

VRTK: A cross platform tool that provides common features to VR projects like teleporting, interactable objects and controller interactions.

Resource Page

Basic syntax and common Unity API functions

Lesson 1:

- https://medium.com/@astarteny/team-building-with-improv-games-c63ae0e02661 A list of improv style icebreaker games

- https://unity3d.com/learn/tutorials/topics/interface-essentials/interface-overview Video tutorial of the Unity interface
-
- https://unity3d.com/learn/tutorials/topics/interface-essentials/interface-overview Overview of the Unity interface

Lesson 2:

- https://docs.unity3d.com/Manual/Prefabs.html
- https://unity3d.com/learn/tutorials/topics/physics/rigidbodies

Lesson 3:

- http://www.openxcell.com/need-precise-game-design-document/ An overview of GDDs

Lesson 4:

- https://www.youtube.com/watch?v=UvCri1tqlxQ Short video on the concept of MVP from Extra Credits
- https://youtu.be/8xAPps2e3xs A video showing iPhone prototypes

Lesson 5:

- https://docs.unity3d.com/Manual/ScriptingSection.html A resource to learn the basics of how scripts work in Unity. Not the best for beginners.
- https://www.raywenderlich.com/227-unity-cheat-sheet-and-quick-reference-2018 Quick reference for common Unity scripting features

Lesson 6:

- http://www.getty.edu/education/teachers/classroom_resources/curricula/landscapes/lesson02.html A lesson plan exploring fore, mid and background as it applies to paintings
- https://www.oculus.com/medium/learn/ A collection of videos from the Oculus Medium team that shows how to use the software

Final Lesson

- https://docs.unity3d.com/Manual/PublishingBuilds.html