CS 498: Virtual Reality

Assignment 1: Unity3D Basics

Due: February 11th @ 12:30 PM

In this assignment you will accomplish the following: you will gain some VR experience by running various demos, then gain some Unity expertise.

Part 1

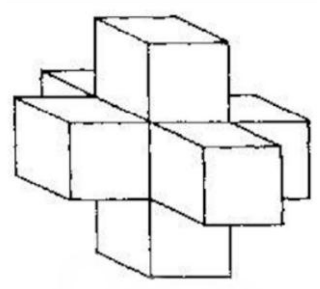
Your first task is to choose and try out three (3) of the VR demos available on the lab machines on the V:\ drive. For each demo, write 2 to 3 sentences including: a short description of the demo, something you liked and something you did not like.

Last, we would like you to find and suggest another VR demo for us to add to our library. You can begin your search here: https://share.oculus.com/. Please include all this information: your choices, your descriptions, and your new demo suggestion, in a PDF named HW1DemoWriteUp.pdf. Submission is covered in the submission section of this document.

Part 2

Part 2 of the assignment is meant to familiarize you with the Unity3D game engine. You will learn the basics of creating and lighting a scene, manipulating GameObjects, using prefabs, and writing scripts. You'll create a simple environment, detailed below.

Room: The room should be constructed out of six **cubes** (one for each surface) and should be exactly 10x10x10 scale units large. Every inner surface of the room (walls, ceiling, floor) must have a solid color (not black or white). To add color and other properties to game objects, you must first create a Material asset and apply it to your object. (Assets -> Create -> Material)



Lighting: Place a point light in the center of the room that casts soft shadows (you must enable shadows by modifying the Shadow Type variable on the point light GameObject using the Inspector).

Stimuli: You will place three stimuli in the room.

1) One sphere of scale 1x1x1 floating in mid-air.
2) One sphere of scale 0.5x0.5x0.5 floating in mid-air next to sphere 1. Make this sphere a child of sphere 1 in the hierarchy.
3) One sphere of scale 0.5x0.5x0.5 floating in mid-air anywhere in the room. This sphere should have a Rigidbody component added (in the Inspector) to it. Be sure to disable gravity on the Rigidbody so that the sphere floats.

All spheres should have a solid color (not black or white).

Scripting: Write a script ('Lightswitch') that turns the point light on or off (either by activating the GameObject or increasing the light intensity) by pressing Tab.

Write a script ('Orbit') that rotates sphere 1 in place and rotates sphere 2 around sphere 1. (Hint: When a transformation is applied to a parent GameObject, all child GameObjects are transformed as well. The parent-child relationship is an important concept to understand, please watch this video: https://goo.gl/oP3c0E)

First, watch this video to learn about triggers: https://goo.gl/F2Nlxd .
Now, create an empty GameObject named "Trigger" and add a Box Collider to it (Inspector -> Add Component -> Physics -> Box Collider). Place the trigger under the sphere and make sure the "Is Trigger" checkbox is checked. Write a script ('Fall') that causes sphere 3 to fall to the floor once the player walks into the trigger.

Player: Place an OVRPlayerController prefab (Assets/OVR/Prefabs/OVRPlayerController) in the room, so that all 3 spheres are visible when the scene starts. The character should not be able to walk through any walls. Rotation and position tracking are enabled by default on the OVRPlayerController prefab.

UI: Add a simple text UI GameObject to your scene (GameObject -> UI -> Text). Change the Render Mode option on the Canvas to “World Space”. The text and canvas will be huge. Resize the canvas using the Scale options (not Width/Height, which is the resolution of the canvas) and align it so that it fits on one of the interior walls of the room. Now, change the text to include your names, a list of the controls, and anything else you want.

Extra Credit: Write a script (‘ProximityDimmer’) that modifies the intensity of the point light based on player distance from the center of the room. The point light becomes brighter when the player approaches the center of the room, and it becomes dimmer when the player moves away from the center. The lightswitch should still function.

Use properly named parent and child GameObjects to keep your scene organized. When you are done, the only top-level objects visible in the Hierarchy should be Room, Stimuli, Player, UI, and Lighting.

Tips and tricks that will make your time with Unity easier:

- Hold V and drag a GameObject corner to snap-align it with other GameObjects.
- Ctrl + D will copy GameObjects.
- Ctrl + Shift + N will create an empty GameObject. You can use this as a parent object.
- Pressing F when an object is selected will center the scene view over that object.
- Separating the Game view and Scene view so both are visible concurrently will make it easier to debug your scene in play mode.
- Use Debug.Log(   ) to print variables and messages to the console when debugging scripts.
- Public global variables in scripts will show up in the Inspector view of the editor.
- The tutorials and documentation on Unity’s website are **extremely** helpful. Going over some of these will make this class a lot easier: https://unity3d.com/learn/tutorials .

Rubric:

Part 1	30%	Answered questions about 3 demos, added suggestion.
Room	10%	Correctly sized room made entirely out of cubes, is colored.
Stimuli	10%	Correct stimulus cube size, position, is colored.
Player	10%	All 3 spheres are visible at start of scene. WASD movement. Can't walk through walls.
Lighting	10%	Proper placement of lighting source.
UI	10%	Text is aligned on wall and properly sized. Contains names and controls.
Scripting	20%	Pressing tab enables/disables the point light. Sphere 2 orbits sphere 1. Sphere 3 falls to ground when trigger is entered.
Extra Credit	10%	Point light brightens and dims with player distance from the center of the room.

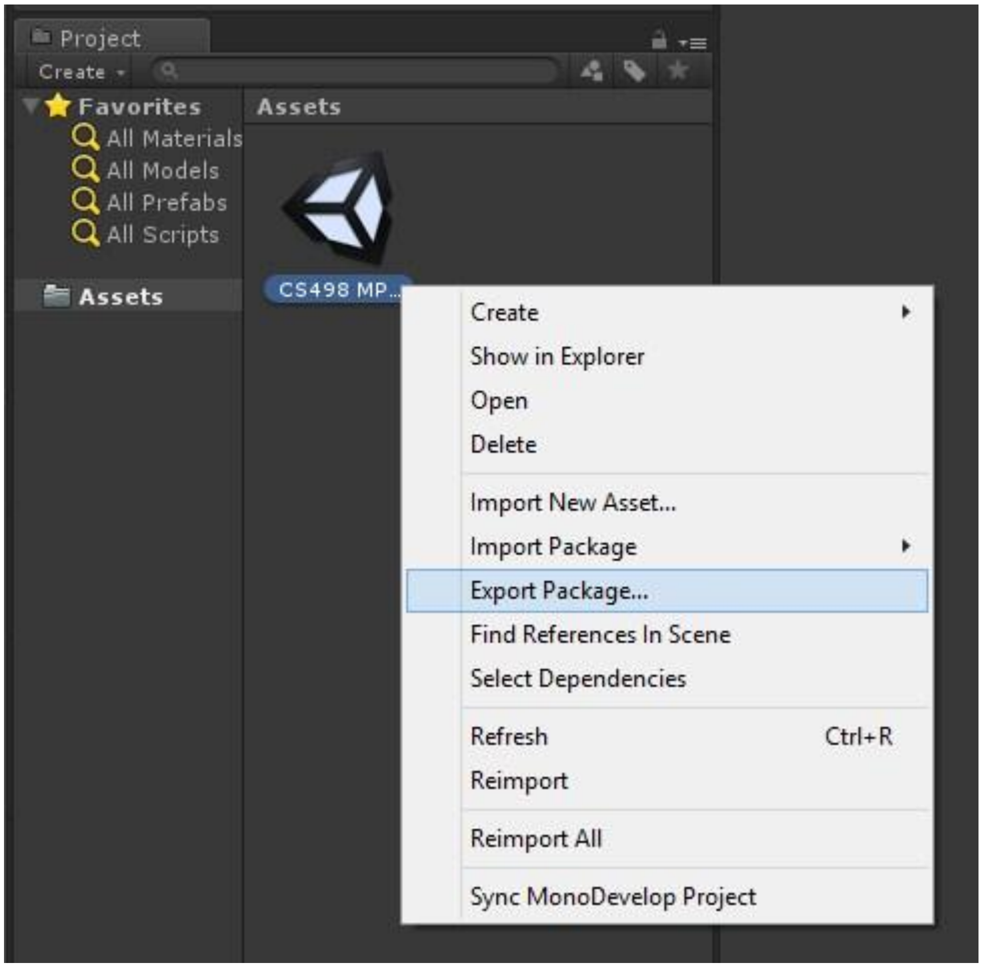
How to Submit the Assignment

Step 1: Create a .unitypackage file

1) Save your Unity scene in the Assets folder with the title “CS498HW1”

2) Using the editor, find the created scene in the Project menu

3) Right click on the scene and select Export Package…



4) Export the file using default settings (“Include dependencies” should be checked by default)

Step 2: Create a standalone game build

1) Go to Edit → Project Settings→ Player. Make sure the "Virtual Reality Supported" box under Other Settings is checked.
2) Go to File → Build Settings
3) Click "Add Current". This will add the current scene to the build. You must have saved the scene to the Assets folder for this to work (you should do that anyways).
4) Hit "Build". Save the project to C:\Users\student's netid\project name, rather than your networked folder.
5) This should create an executable (.exe) for running the build, as well as a folder containing your scene data. Make sure this executable runs correctly on the Rift before submitting.

Step 3: Zip the files and submit them through Compass

1) Create a zip file containing 4 items:
   a) The writeup PDF
   b) The .unitypackage created in Step 1
   c) The .exe and data folder created in Step 2
   d) A README.txt file containing any special instructions or notes you think are relevant for evaluating your assignment.
2) Name the file by separating NetIDs with underscores._cs498sl_HW1.zip
   EXAMPLE: If john1 and carmack2 worked together, the file should be called john1_carmack2_cs498sl_HW1.zip

DO NOT SUBMIT YOUR ENTIRE PROJECT FOLDER