CS 498 Virtual Reality

Assignment 2. Transformations and Depth Perception

Released: September 19th

Due: This MP has four parts, and the deadlines are:

Part 1	September 26th, 11:59 PM
Part 2, Part 3, and Part 4	October 3rd, 11:59 PM

Make sure you read the document all the way through, including submission instructions, before turning it in! You must include the data folder in your submission!

Overview

This assignment will demonstrate the difference between rotation and position tracking. It will help you appreciate how much depth perception relies on both of these, and teach you how to enable and disable them. It'll also help you understand the rigid body transformations covered in class.

We have provided you with a .unitypackage containing a scene ('Assignment 2'). The provided scene contains a MainCameraParent prefab, which doesn't allow player movement but still uses CV1's rotation and position trackers to modify camera position and rotation. Make sure you have the "Virtual Reality Supported" option checked under Edit → Project Settings→ Player in order to test out your project on the CV1.

Note: If this is your first time using Oculus Runtime, you need to sign up for a free Oculus account and go through several steps including adjusting the sensor and IPD to set up the headset. Oculus Runtime will provide clear instructions, but feel free to ask questions if you encounter any difficulty.

Part 1 VR Mirror

This part of the assignment will familiarize you with manipulating a GameObject's rotation and position. Tasks that need to be completed in this part include:

- Pressing tab should reset the main camera position in global coordinates to (0,0,0).
- Pressing the F key should flip the user 180 degrees so that he is looking behind where he was looking.
- Pressing the M key should make a certain cube either mirror or follow the user's movements.
- Pressing the Esc key should quit.

Create a script named CameraReset that moves the Main Camera to the position (0,0,0) in global space when you press the tab key. Pressing tab multiple times should move the camera back to the origin each time, regardless of where the player moves their head between presses. Note: “global space” refers to the sum of the Main Camera and the parent.

Create a script named CameraFlipper that flips the MainCameraParent 180 degrees when you press the F key, as if you were turning around to look behind you. After flipping the first time, you should be facing a transparent window with a floating cube face on the other side. You should also be able to press the F key multiple times to keep flipping 180 degrees. CameraReset should remain functional regardless of orientation.

Write a script VRMirror to modify the position and rotation of the cube face to match or mirror your own. The face should start out neither matching nor mirroring Pressing the M key once should make the cube match your movements. This includes positional and rotational movements . Furthermore, note that this also means that the cube should be looking in the same direction the camera is (and as result, the user should see the back of the cube’s head). For example, if you bring your face closer to the screen, the cube moves further away from the screen.

Pressing the M key again should make the cube mirror your movements (positional and rotational). This means that the cube should facing the camera when you are looking into the mirror (and as a result, the user should see the cube’s face). In this case, if you bring your face close to the screen, the cube moves closer to the screen as well. Imagine looking into a mirror to get an intuition of this.

Pressing the M key after this should switch between matching and mirroring

The cube’s behaviour before the first M keypress is your choice and we won’t be checking for that. You can even make the cube not move at all if you wish to.

Note: The “forward” axis of the mirror-cube is 90° offset from that of the camera - you will need to account for this in your solution!

Part 2 Disabling Position and Rotation Tracking

Now you will figure out a way to disable position and rotation tracking. Create a script named ToggleTracking that turns tracking on and off. Tasks that need to be completed in this part include:

- Pressing the R key should toggle rotation tracking on and off.
- Pressing the P key should toggle position tracking on and off.

Temporarily losing rotation and position tracking in this Assignment should help you understand how much they contribute towards presence in VR.

Note 1: One of the bad solutions is to put everything as the children of the camera. This solution depends on the hierarchy structure of Unity and causes performance issues and hence we will not accept it. We want you to think about how to reverse/counteract the effects of position and rotation tracking using transformations.

Note 2: When disabling rotation and translation, you can assume that we won't test both tasks together, i.e, you don't need to disable position and rotation tracking at the same time. We will give 5% extra credit if you do figure out how to disable position and rotation tracking together. Please leave a note in your readme.txt if you complete the extra credit.

Part 3 Depth Perception and Relative Size

In your scene, there are three spherical gameobjects with color red and blue that are children of the parent gameobject called StimulusManager. Put the red sphere at position (0, 0, -2.3) and the two blue spheres at positions (0.7, 0, -1.6), (-0.8, 0, -1), respectively. Create a script called GenerateStimuli that dynamically changes the sizes of the two blue spheres so that no matter what the position and orientation of the camera are, the blue spheres always “appear” to be at the same depth as the red sphere with respect to their retinal image sizes (reference: http://vr.cs.uiuc.edu/vrch6.pdf Section 6.1.1). That is, we ask you to vary the sizes of the blue spheres so that the **rendered** images of all of the three spheres have equal diameters when viewed on the Oculus Rift display:

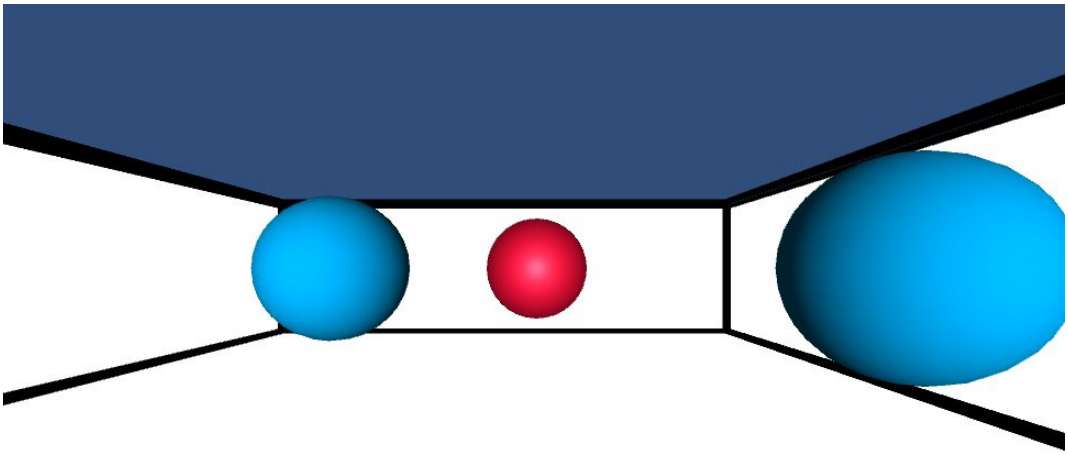

Figure 1: Default scene display

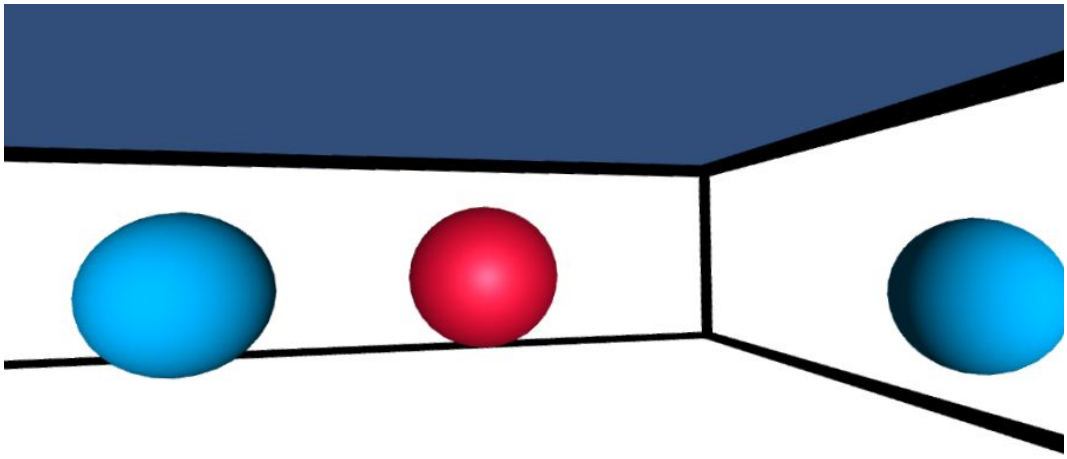

Figure 2: After GenerateStimuli activates

Please do not change neither the positions of the three spheres nor the size of the red sphere. You are only allowed to change the sizes of the blue spheres to complete this task.

Other tasks that need to be completed in this part are:

- Pressing the S key should toggle between appearance and disappearance of the three spheres. They should appear(disappear) in the manner described below.
- The red sphere should appear(disappear) immediately after the user presses the S key.
- The two blue spheres should appear(disappear) exactly two seconds after the red sphere appears(disappears). You may find the documentation for Time.deltaTime useful: https://docs.unity3d.com/ScriptReference/Time-deltaTime.html .
- You can assume that the user will not press S key within the two seconds.

The script GenerateStimuli should activate when the user presses the S key for the first time.

Note: You should understand from this task that you can easily infer the relative depth between the three spheres even after you are finished with the task because there are many other depth cues other than retinal image size that help you perceive depth accurately. Therefore, this exercise will not create an illusion of the spheres being placed at equal distances from your eyes. You will learn more about optical illusions and depth perception later in class.

Part 4 Written assignment

Complete the following problems. Only electronic submissions are accepted.

1. In one sentence, explain what the following homogeneous transformation accomplishes when applied to a point (x, y, z), in terms of yaw, pitch, roll, and translation.

$$T_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2. Write out the 4x4 homogeneous transformation T_2, when applied to a point (x, y ,z) in global coordinate frame, translates the point by $(3, 0, 2)^T$, then followed by a pitch of 45 degrees. Your answer need not be simplified, and may be represented as a single matrix or the product of two or more matrices.
3. We would like to reverse the transformation applied by $T_2\,T_1$, that is, write out $(T_2 T_1)^{-1}$. Your answer need not be simplified, and may be represented as a single matrix or the product of two or more matrices.
4. Write out the Quaternion equivalent to the rotations in T_1 and T_2 as q_1 and q_2. Then calculate the product, that is, $q_1 \circ q_2$ (Hint: Steve's book may be a good source of reference)

Rubric

Criteria Name	Points	Description for full credit
VR Mirror 1	10%	Pressing tab resets the camera position to (0,0,0). Pressing F flips the camera 180 degrees.
VR Mirror 2	10%	Cube correctly matches your movements.
VR Mirror 3	10%	Cube correctly mirrors your movements.
VR Mirror 4	5%	Pressing the M key toggles the cube between mirroring and matching your movements.
Disable Tracking 1	20%	Pressing the R key toggles rotation tracking on and off.
Disable Tracking 2	5%	Pressing the P key toggles position tracking on and off.

Depth perception 1	10%	Objects appear in the correct order, following the correct timing. The script can be run repeatedly
Depth perception 2	10%	Objects appear to be of equal size once the script has been run. (not precomputed)
Written Assignment	20%	All computations are correct; show work.
Extra Credit	5%	Disable position and rotation tracking at the same time

Submitting

Follow the steps described below for submission.

NOTE: Assignments must be completed in groups of 2.

Step 1: Create a .unitypackage file

1) Save your Unity scene in the Assets folder with the title “CS498HW2”

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, a UnityPlayer.dll, as well as a folder containing your scene data. Make sure this executable runs correctly on the Rift before submitting.

Step 3: Create a PDF file for the written assignment

1) Write out the written assignment using LaTeX, word or editor of your preference.

2) Export the file to PDF format. Name the file in the format of “hw2.pdf”

3) Only one written assignment needs to be submitted for each group

Step 4: Zip the files and submit them through Compass

1) Create a zip file containing 3 items:

   a) The .unitypackage created in Step 1

   b) The .exe AND data folder AND .dll created in Step 2

   c) The .pdf file created in Step 3 (You can skip this step when submitting part 1 and part 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_HW2.zip EXAMPLE: If john1 and carmack2 worked together, the file should be called john1_carmack2_cs498sl_HW2.zip

DO NOT SUBMIT YOUR ENTIRE PROJECT FOLDER