Spring 2021

CSE 190 VR Technologies

Discussion 7





ANNOUNCEMENTS

- Homework 3 Deadline Extended for A WEEK
 - Due Sunday (5/23)
 - \circ Come to Office Hours
- Homework 4 Released
 - Due Sunday (5/30)
 - START EARLY





AGENDA

- Homework 4 Overview
- Homework 4 Getting Started
- Homework 3 Q&A





Homework 4 Overview

m

Homework 4 Overview

- Simulate the CAVE (Cave Automatic Virtual Environment) in VR (A virtual environment within a virtual environment...?)
- Simulate the tracked user
- Simulate an observer



Homework 4 Overview - View Point Switch

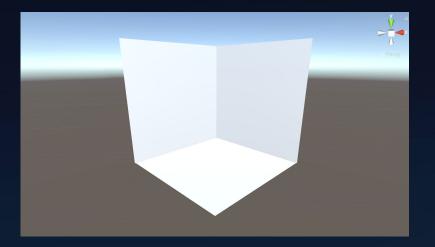
- HMD View & Controller View
 - \circ ~ Able to switch from HMD view to Controller View
 - HMD view: view point same as headset
 - Controller view: view point same as right controller
- Freeze View
 - Freeze the view point that is used to render to CAVE wall
- Debug View

<u>Homework 4</u> <u>Getting Started</u>



Homework 4 Getting Started: Create CAVE

- Create CAVE Room (Three 3D Planes)
 - Can use GameObject -> Plane



Render to CAVE Walls

- You need two pairs of cameras (each pair for left/right eyes)
 - One pair looks at the CAVE wall (OVRCameraRig)
 - The other pair serves as the view point to the virtual scene (Custom Camera Rig)
 - CAVE walls should be **invisible** to the **custom rig**
- Render the image from the view point (seen by the custom rig) to the CAVE walls use off-screen rendering
- Needs a custom material and a custom shader for the walls in order to display off-screen rendering images



Render to CAVE Walls: Invisibility

- Don't want the view point camera see the CAVE walls
- Set the Render Layer of the CAVE walls

Inspector		a :
∭_ ✓ ImagePlane	Left	🗌 🗌 Static 🔻
Tag Untagged	Layer CAVE	•
🔻 🙏 🛛 Transform		0 ∓ :
Position	X -1.56851 Y 1.562999	Z -4.84633
Rotation	X 90 Y 90	Z -45
Scale	X 0.5 Y 0.5	Z 0.5
🔻 🆽 🛛 Plane (Mesh P	ilter)	;



20

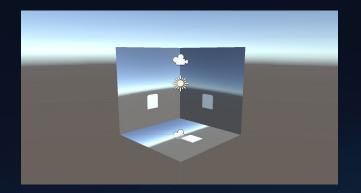
Render to CAVE Walls: Invisibility

- Set the render layer for the virtual scene objects
- Set the Culling Mask for both pair of cameras
 - The CAVE-looking cameras should see all CAVE walls, but should not be able to see scene objects
 (OVRCameraRig -> Culling Mask -> Exclude Scene)
 - The scene-looking cameras should NOT see any CAVE walls, but can see all scene objects (CustomCameraRig -> Culling Mask -> Only Include Scene)

:ọ: 🔺	Account	Layers			0: Default
 Inspector 					1: TransparentFX
	ube				2: Ignore Raycast
	Intagged	-	Lav	or	4: Water
ray C	maggeu		Lay	CI	5: UI
🔻 🙏 🛛 Tra	ansform				6: CAVE
Position		0		2.	✓ 7: Scene
Rotation				4:	Add Layer
Scale		3.8984		3.	Adu Layei

🗖 🗹 Camera		0	
Clear Flags	Skybox		
Background			1
Culling Mask			
Projection	Nothing		
FOV Axis	Everything		
Field of View	Default		ĥ
Physical Camera	TransparentFX		
	Ignore Raycast		
Clipping Planes	Water		
	UI		
Viewport Rect	CAVE		
	✓ Scene		

- Need to render the view from the custom camera rig to the CAVE walls
- Need to create material with custom shader
- Modify the material shader from Homework 3 (material.shader)



```
Properties
Properties
                                                                              \_Color ("Color", Color) = (1,1,1,1)
    \_Color ("Color", Color) = (1,1,1,1)
                                                                              _MainTexLeft ("Left Texture", 2D) = "white" {}
    _MainTex ("Albedo (RGB)", 2D) = "white" {}
                                                                              _MainTexRight ("Right Texture", 2D) = "white" {}
    _Glossiness ("Smoothness", Range(0,1)) = 0.5
                                                                              _Glossiness ("Smoothness", Range(0,1)) = 0.0
    _Metallic ("Metallic", Range(0,1)) = 0.0
                                                                              Metallic ("Metallic", Range(0,1)) = 0.0
                                                                          }
                                                                           // Use shader model 3.0 target, to get nicer looking lighting
// Use shader model 3.0 target, to get nicer looking lighting
                                                                           #pragma target 3.0
#pragma target 3.0
                                                                          sampler2D _MainTexLeft;
sampler2D _MainTex;
                                                                           sampler2D _MainTexRight;
struct Input
                                                                           struct Input
                                                                              float2 uv_MainTexLeft;
    float2 uv_MainTex;
                                                                              float2 uv_MainTexRight;
                                                                           }:
```

```
void surf (Input IN, inout SurfaceOutputStandard o)
                                                                               // Albedo comes from a texture tinted by color
void surf (Input IN, inout SurfaceOutputStandard o)
                                                                               fixed4 c;
{
                                                                               if (unity_StereoEyeIndex == 0) { // Left
    // Albedo comes from a texture tinted by color
                                                                                   c = tex2D(_MainTexLeft, IN.uv_MainTexLeft) * _Color;
    fixed4 c = tex2D (_MainTex, IN.uv_MainTex) * _Color;
    o.Albedo = c.rab:
                                                                               else { // Riaht
                                                                                   c = tex2D(_MainTexRight, IN.uv_MainTexRight) * _Color;
    // Metallic and smoothness come from slider variables
    o.Metallic = _Metallic;
                                                                               o.Albedo = c.rgb;
    o.Smoothness = _Glossiness;
                                                                               // Metallic and smoothness come from slider variables
    o.Alpha = c.a;
                                                                               o.Metallic = _Metallic;
                                                                               o.Smoothness = _Glossiness;
                                                                               o.Alpha = c.a;
```

- This shader ensures that different set of images are rendered to the CAVE walls, corresponding to left and right eyes
- Create a new material and attach this shader

- Create two Render Textures for the material
 - Assets -> Create -> Render Texture
 - Create 2, one for left eye, one for right eye
- Select higher size for the texture (e.g. 1024x1024)
- Attach this material to all three CAVE planes

Inspector			a :
Render Texture	(Render Textur	e)	0 ‡ ;
			Open
Dimension	2D		•
Size	1024	x 1024	
Anti-aliasing	2 samples		•
Enable Compatible Co	~		
Color Format	R8G8B8A8_UNO	RM	•
Depth Buffer	At least 24 bits d	epth (with ste	ncil) 🔻

🔁 Ins	spector				a :
	Stereo 7	Fexture	(Material		0∓:
	Shader	Project2	2/material		▼ Edit
Col	or				28
Lef	t Texture				
٦	Tiling	X 1		Y 1	
C	Offset	X 0		Y 0	Select
Rig	ht Texture				
Г	Tiling	X 1		Y 1	
C	Offset	X 0		Y 0	Select
Sm	oothness		•		0
Me	tallic		•		0
Rer	nder Queue	e		From Shader	v 2000
	ble GPU Ir		a		
	uble Sided		0		
		orobur n	annation		

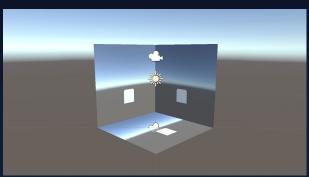
- Download the Off-Screen Rendering Script
 - <u>https://gist.github.com/danielbierwirth/10965844fecc38243007f0cd21843d90</u>
- Create an empty GameObject, and attach the script to it
- Set correct camera correspondences (remember to select the cameras

from the custom camera rig, which looks to the scene)

	:
Script 🛛 🖩 OffscreenRendering 🖉	•
Screenshots Per Second 1	
Offscreen Camera Left 🛛 OffscreenCameraLeft (Camera) 🔗	•
Offscreen Camera Right None (Camera)	•

- Set the scene-looking cameras' target texture to the newly created textures
- Now you should be able to see something like this:





🗖 🖬 🖌 Camera		9 ∓±	:
Clear Flags	Skybox		•
Background			de.
Culling Mask	Scene		•
Projection	Perspective		•
FOV Axis	Vertical		•
Field of View	•	60	
Physical Camera			
Clipping Planes	Near 0.3		
	Far 1000		
Viewport Rect	X 0 Y 0		
	W 1 H 1		
Depth	0		
Rendering Path	Use Graphics Settings		•
Target Texture	RenderTexture		\odot
Occlusion Culling			
HDR	Use Graphics Settings		•
MSAA	Use Graphics Settings		•
Allow Dynamic Resolution			
Target Display	Display 1		•
Target Eye	None (Main Display)		•

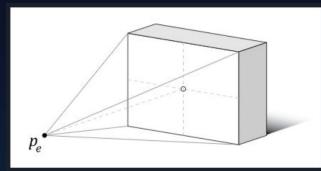
RenderTexture currentRT = RenderTexture.active; RenderTexture.active = texture; camera.targetTexture = texture; Matrix4x4 origP = camera.projectionMatrix; camera.projectionMatrix = P; camera.Render(); // Read offscreen texture Texture2D offscreenTexture = new Texture2D(texture.width, texture.height, TextureFormat.RGB24, false); offscreenTexture.ReadPixels(new Rect(

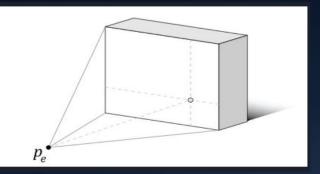
```
0,
0,
texture.width,
texture.height
), 0, 0, false);
```

offscreenTexture.Apply(); RenderTexture.active = currentRT;

Render to CAVE Walls: Projections

- Reminder a typical projective matrix assumes we are right in front of the screen
- We need to be able to render off-center

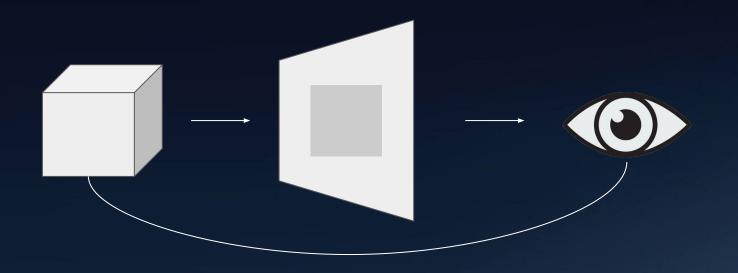




Render to CAVE Walls: Projections

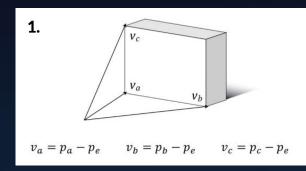
• Review of the projection matrices

$$P' = PM^TT$$



Render to CAVE Walls: Projections - P

- 1. Calculate vectors from eye position to the screen corners
 - o plane.GetComponent<Renderer>().bounds.max;
 - o plane.GetComponent<Renderer>().bounds.min;
- 2. Calculate distance from eye position to screen space origin



$$2. \quad d = -(v_n \cdot v_a)$$

$$P' = PM^TT$$

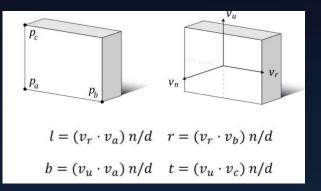
Render to CAVE Walls: Projections - P

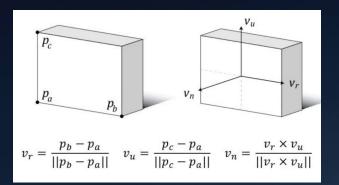
3. Calculate the frustum extents at the near plane

 P = Matrix4x4.Frustum(float left, float right, float bottom, float top, float zNear, float zFar);

$$P' = PM^TT$$

- Near and far define the near/far clipping plane
 - Depends on how you want to clip user's view





Render to CAVE Walls: Projections - M

- We want to transform the screens XY plane to be aligned with the viewer XY plane
- M: maps into screen coordinates
- Want to go from screen coordinates to viewer so we take the inverse of M and get M⁻¹ = M^T
- Note that Unity Matrix is COLUMN MAJOR

$$M^{T} = \begin{bmatrix} v_{rx} & v_{ry} & v_{rz} & 0\\ v_{ux} & v_{uy} & v_{uz} & 0\\ v_{nx} & v_{ny} & v_{nz} & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = PM^TT$$

Render to CAVE Walls: Projections - T

• Pe: Position of scene-looking camera

$$P' = PM^TT$$

$$T = \begin{bmatrix} 1 & 0 & 0 & -p_{ex} \\ 0 & 1 & 0 & -p_{ey} \\ 0 & 0 & 1 & -p_{ez} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Render to CAVE Walls: Projections

- With P', you can set the projection matrix of the scene-looking cameras
 - o camera.projectionMatrix = pPrime;
- The off-screen render script will handle the rest!
- Remember to track the headset pose for your custom camera rig!
 - camParent.transform.localPosition =
 UnityEngine.XR.InputTracking.GetLocalPosition(UnityEngine.XR.XRNode.LeftEye
);

 $P' = PM^TT$

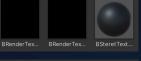
Render to CAVE Walls: Three Walls

- Now you should be able to render the same image to all three walls
- However we want to render different images to the three walls, as we have three different off-center projections
- Need the following modifications:
 - No need to set camera target texture any more, set it in code, point to the following textures
 - Create two more materials with stereo textures
 - LeftWallMaterial (Already have)
 - RightWallMaterial
 - RightWallLeftEyeTexture
 - RightWallRightEyeTexture
 - BottomWallMaterial
 - BottomWallLeftEyeTexture
 - BottomWallLeftEyeTexture

Depth		
Rendering Path	Use Graphics Settings	
Target Texture		
Occlusion Culling	~	
HDR	Use Graphics Settings	
MSAA	Use Graphics Settings	

public RenderTexture leftPlaneLeftTexture; public RenderTexture rightPlaneLeftTexture; public RenderTexture bottomPlaneLeftTexture; public RenderTexture leftPlaneRightTexture; public RenderTexture rightPlaneRightTexture; public RenderTexture bottomPlaneRightTexture;

🔻 # 🗹 Offscreen Renderii	ng (Script) 🛛 🥹 👎	
	# OffscreenRendering	
Screenshots Per Second		
Offscreen Camera Left	OffscreenCameraLeft (Camera)	
Offscreen Camera Right	OffscreenCameraRight (Camera)	
Left Plane	© ImagePlaneLeft	
Right Plane	☺ ImagePlaneRight	
	🕆 ImagePlaneBottom	\odot
Left Plane Left Texture	₿LRenderTextureLeft	
Right Plane Left Texture	₿LRenderTextureLeft	
Bottom Plane Left Texture	◎LRenderTextureLeft	
Left Plane Right Texture	None (Render Texture)	
Right Plane Right Texture	None (Render Texture)	
Bottom Plane Right Texture	None (Render Texture)	







Debug Mode

- Debug Mode to assist you with "head-in-hand" mode
- Visualize the "eye positions" of the controller
- Visualize the pyramids

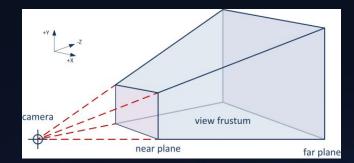
- You need to draw 6 pyramids to both eyes
 - NOT 3 pyramids for each eye
 - Meaning you should see all 6 pyramids in both eyes

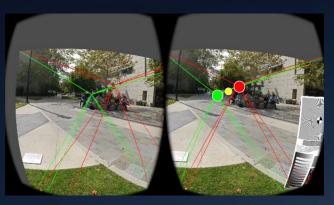
Green Dot: Left Eye Position (On the controller)

Red Dot: Right Eye Position (On the controller)

Yellow Dot: Controller Position (Just for your understanding)

P.S. Those dots don't need to be rendered





Extra Resources

(Can also be found on course website)



Homework 4 Extra Resources

- Offscreen Rendering in Unity (Required to render camera views to the CAVE walls)
 - <u>https://gist.github.com/danielbierwirth/10965844fecc38243007f0cd21843d90</u>
- Off-Center Projection Matrix Calculation
 - <u>https://web.archive.org/web/20190219024806/http://csc.lsu.edu/~kooima/articles/genperspective/</u>
- Original CAVE Paper
 - <u>http://www.cs.utah.edu/~thompson/vissim-seminar/on-line/CruzNeiraSig93.pdf</u>

Homework 3 Q&A

In