# CSE 190: Virtual Reality Technologies 

LECTURE \#14: RENDERING TO HMDS

## Upcoming Deadlines

Sunday, May 16: Project 3 due
Monday, May 17: Discussion Project 4
Sunday, May 23: Project 3 late deadline
Monday, May 24: Discussion Project 4
Sunday, May 30: Project 4 due

## App Presentations

Matthew Engurasoff

- Rhythm Dungeon

Shane Li

- Gorn


# Rendering to <br> CAVE Screens 

REQUIRED FOR PROJECT 4

## Demo Video


https://www.mechdyne.com/av-vr-solutions/solutions/virtual-augmented-reality/cave/

## CAVE Simulator

- Key features to implement:
- Render the scene to 3 squares
- Ability to switch the viewport from HMD position to the Controller position
- Ability to freeze the viewport position
- Manipulate cubes


Generalized Perspective Projection

## Perspective Projection

- Typically we use a symmetrical projection matrix
- This works under the assumption that we are directly in front of the screen, along the center axis
- We are looking at the center of the screen



## Projection Matrix for CAVE Screens

- A typical projective matrix assumes we are right in front of the screen
- But we need to be able to be off-center



## Off-axis Perspective Projection

- In a CAVE, we cannot view every screen head on, so each screen needs a different perspective



## Off-Center Viewing: Example



## Calculating Frustum Parameters

1. Calculate vectors from eye position to the screen corners
2. Calculate distance from eye position to the screen space origin

3. $d=-\left(v_{n} \cdot v_{a}\right)$

$$
v_{a}=p_{a}-p_{e} \quad v_{b}=p_{b}-p_{e} \quad v_{c}=p_{c}-p_{e}
$$

## Projection Matrix for CAVE Screens - P

1. Compute basis vectors for screen space


$$
v_{r}=\frac{p_{b}-p_{a}}{\left\|p_{b}-p_{a}\right\|} \quad v_{u}=\frac{p_{c}-p_{a}}{\left\|p_{c}-p_{a}\right\|} \quad v_{n}=\frac{v_{r} \times v_{u}}{\left\|v_{r} \times v_{u}\right\|}
$$

## Calculating Frustum Parameters

3. Calculate the frustum extents at the near plane


$$
\begin{array}{ll}
l=\left(v_{r} \cdot v_{a}\right) n / d & r=\left(v_{r} \cdot v_{b}\right) n / d \\
b=\left(v_{u} \cdot v_{a}\right) n / d & t=\left(v_{u} \cdot v_{c}\right) n / d
\end{array}
$$

## Almost there

- We need two more capabilities:
- Rotate the screen out of the XY plane
- Correctly position it relative to the user


## Projection Matrix for CAVE Screens

- Review of what we did to get the projection matrix

$$
P^{\prime}=P M^{T} T
$$

- This gives you the projection matrix ( $P^{\prime}$ ) for each screen


## Projection Matrix for CAVE Screens - P

- Now that we have our frustum parameters we can calculate the P matrix by simply calling:
glm: :mat4 P = glm: :frustum(l, r, b, t, near, far);

$$
P^{\prime}=P M^{T} T
$$

## Projection Matrix for CAVE Screens - $\mathrm{M}^{\top}$

- Review of the formula for $\mathrm{M}^{\top}$

$$
M^{T}=\left[\begin{array}{cccc}
v_{r x} & v_{r y} & v_{r z} & 0 \\
v_{u x} & v_{u y} & v_{u z} & 0 \\
v_{n x} & v_{n y} & v_{n z} & 0 \\
0 & 0 & 0 & 1
\end{array}\right]
$$

- We already calculated $v_{r}, v_{u}$, and $v_{n}$
- So all we need to do is create a mat4 for $\mathrm{M}^{\top}$ and plug those vectors in
glm::mat4 $M=$ glm::mat4(vr, vu, vn, glm::vec4(0, 0, 0, 1));


## Projection Matrix for CAVE Screens - T

- Review of the formula for $T$

$$
T=\left[\begin{array}{cccc}
1 & 0 & 0 & -p_{e x} \\
0 & 1 & 0 & -p_{e y} \\
0 & 0 & 1 & -p_{e z} \\
0 & 0 & 0 & 1
\end{array}\right]
$$

Reminder:
$\mathrm{p}_{\mathrm{e}}=$ eye position
$T=$ translation matrix by $-p_{e}$
glm::mat4 T = glm::translate(glm::mat4(1.0f), -p_e);

## Projection Matrix for CAVE Screens

- Now take a look at the formula again

$$
P^{\prime}=P M^{T} T
$$

$P^{\prime}$ is the actual projection that we want to return, not $P$

- What's the next step when I get the projection?
- Draw your scene to your off-screen buffers
- Render them onto the texture of your screen

Viewport Switch

## Viewport Switch

Currently your View position is coming from the Position and Orientation of your HMD

Need to be able to switch the view position to your right controller

- This is simulating being a spectator in a CAVE with another person wearing the head tracker
- Your controller is acting as that person's head


## Viewport Switch

## Note:

- When your rotate your head, the scene on the screens should not rotate
- So when you rotate your controller in this mode, the scene should not rotate
- You still have two "eyes" on your controller in this mode


## Viewport Switch

Although rotation is not reflected in the scene, you are still expected
to see some changes while rotating controller:

- Controller's forward direction is perpendicular to your head forward direction
- The image becomes mono
- Controller's forward direction is in reverse direction
- The image is inverted stereo


## Resources

- Article on off-center viewing matrices:
- https://web.archive.org/web/20190219024806/http://csc.Isu.e du/~kooima/articles/genperspective/
- SIGGRAPH paper on CAVE projection:
- http://www.cs.utah.edu/~thompson/vissim-seminar/online/CruzNeiraSig93.pdf

SMP

## NVIDIA SMP (Simultaneous MultiProjection)



Up to 16 independent viewports can be projected simultaneously in one rendering pass

- Includes stereo (=2 viewports)

Video (1'50+): https://www.youtube.com/watch? $\mathrm{v}=\mathrm{p} 6 \mathrm{NbyEmPalA}$


# Display Limitations 

## Lenses for VR HMDs



How lenses for VR HMDs work:

- https://www.youtube.com/watch?v=NCBEYaC876A



## Focal Distance

Apparent distance from eye to where the pixels are in focus.

| HMD | Focal Distance |
| :--- | :--- |
| Oculus DK1 | Infinity |
| Oculus DK2 | 1.4 meters |
| Oculus CV1 | 2 meters |
| Oculus Quest 1 and 2 | 2 meters |
| HTC Vive, Vive Pro | $\sim 1$ meter |
| Valve Index | Infinity |

## Lens Distortion

All VR HMDs have lenses which distort the image.
VR engine has to render a pre-distorted image so that the user will see a correct, undistorted image. A simple pixel shader can do this.

Barrel Distortion (In-Engine)


Pin-cushion Distortion (From Rift Lenses)


No Distortion
(Final Observed Image)


## Lens Distortion



## Screen Door Effect

Because pixels on LCD and OLED displays have dead space in-between them image looks like looking through a screen door when looking at it through magnifying lenses.


LCD
DK1


OLED
DK2


Screen Door

## Chromatic Aberration

Arises from the inability of a lens to focus all colors in the same place.

FOcal length depends on refraction.
blue and red light have different indexes of refraction $\rightarrow$ their focal length is also slightly different.

Chromatic aberration is clearly visible on
 photographs or video as the color channels are not perfectly aligned.

Remedy: apply "Brown's model" distortion correction formula to each color channel independently.

