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 CAVE Screens

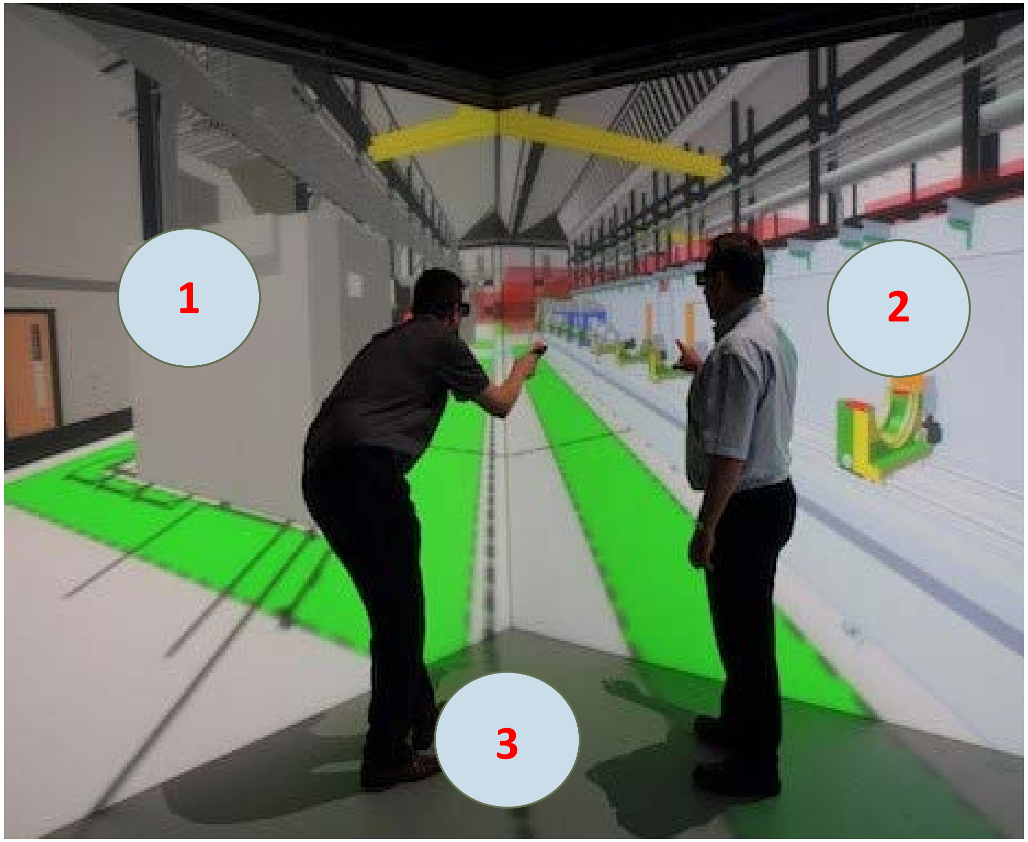
REQUIRED FOR PROJECT 4

Demo Video

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

CAVE Simulator

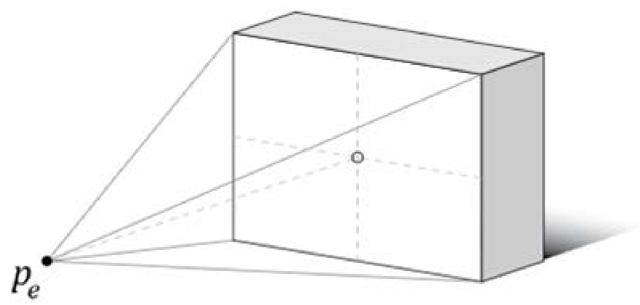
- Key features to implement:
  - Render the scene to 3 squares
  - Ability to switch the viewport from HMD position to the Controller position
  - Ability to freeze the viewport position
  - Manipulate cubes



Generalized Perspective Projection

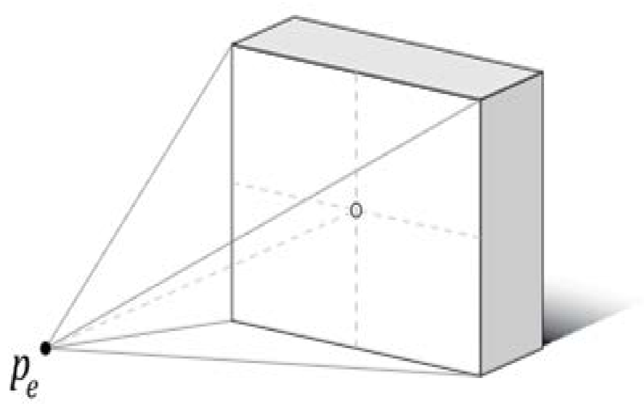
Perspective Projection

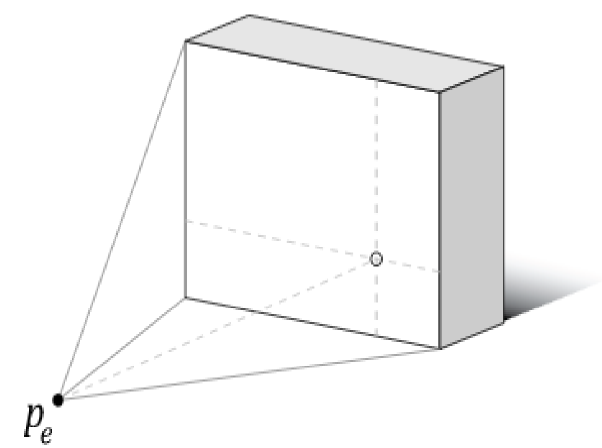
- Typically we use a symmetrical projection matrix
- This works under the assumption that we are directly in front of the screen, along the center axis
- We are looking at the center of the screen



Projection Matrix for CAVE Screens

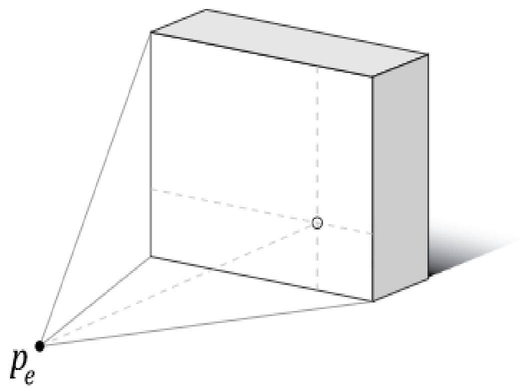
- A typical projective matrix assumes we are right in front of the screen
- But we need to be able to be off-center

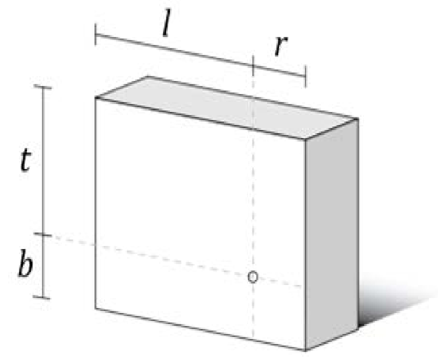




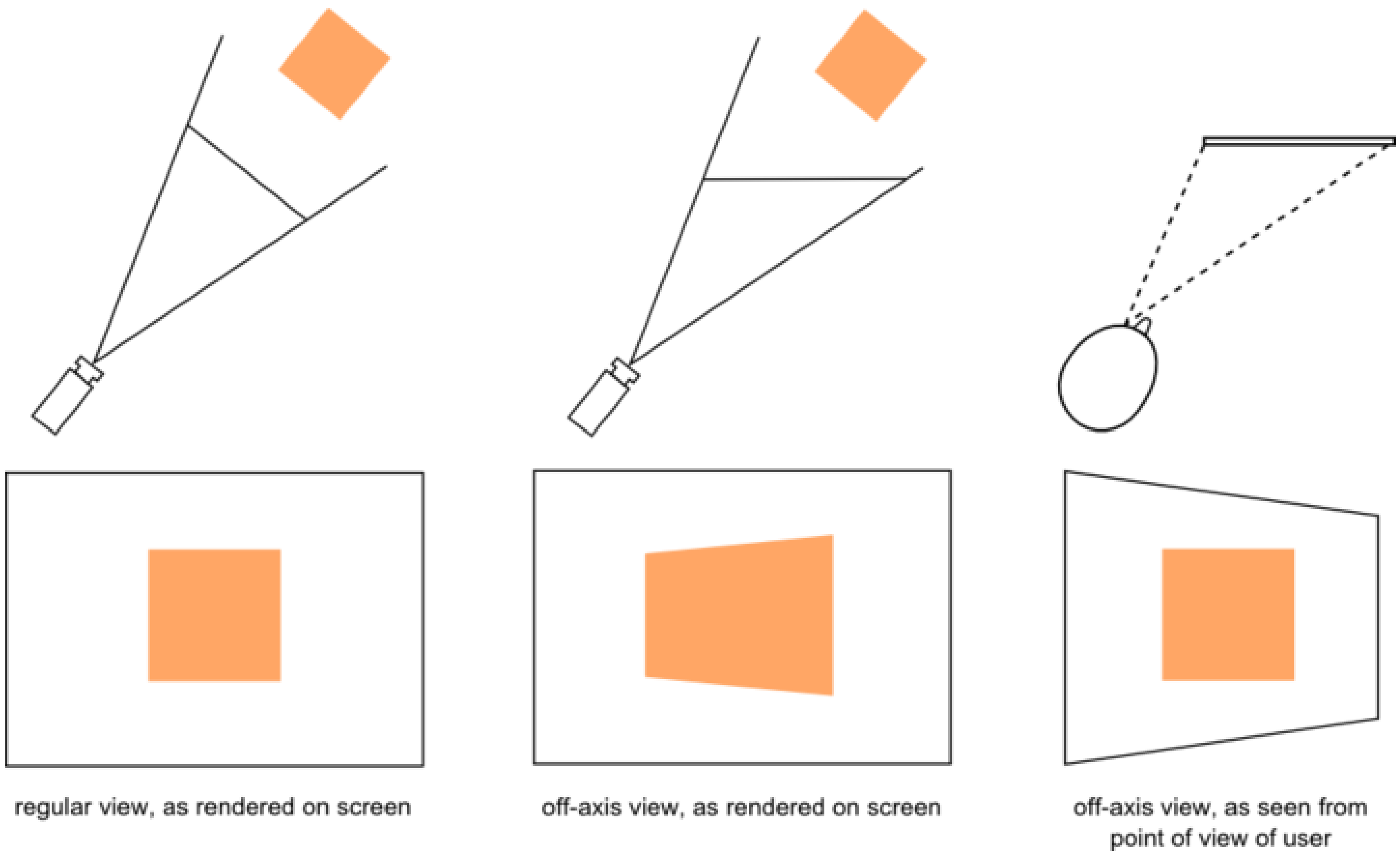
Off-axis Perspective Projection

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



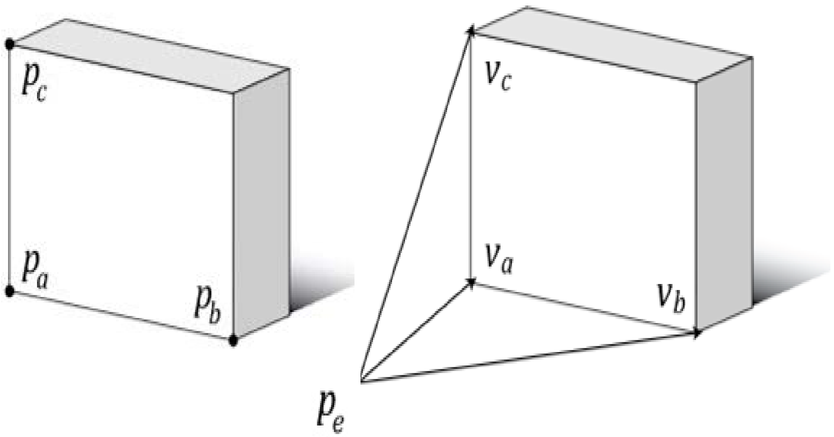


Off-Center Viewing: Example



Calculating Frustum Parameters

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

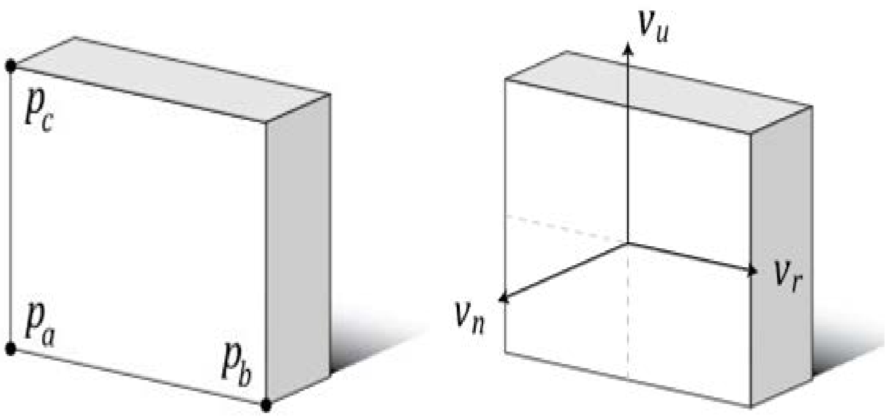


2. $d = -(v_n \cdot v_a)$

1. $v_a = p_a - p_e \qquad v_b = p_b - p_e \qquad v_c = p_c - p_e$

Projection Matrix for CAVE Screens - P

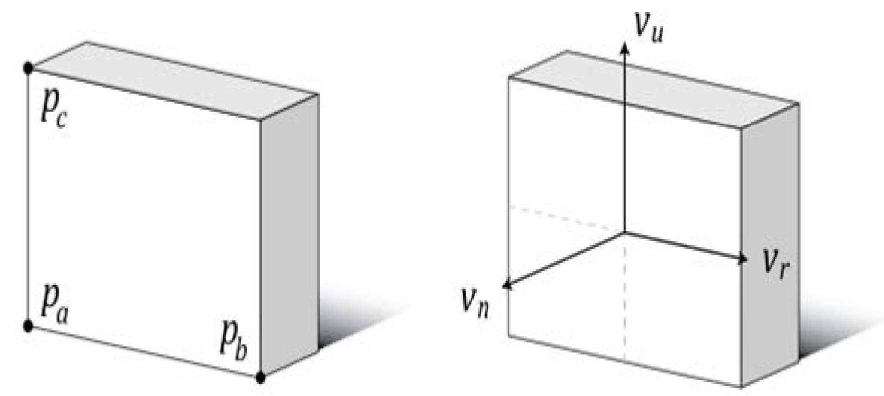
3. Compute basis vectors for screen space



$$v_r = \frac{p_b - p_a}{||p_b - p_a||} \quad v_u = \frac{p_c - p_a}{||p_c - p_a||} \quad v_n = \frac{v_r \times v_u}{||v_r \times v_u||}$$

Calculating Frustum Parameters

4. Calculate the frustum extents at the near plane



$$l = (v_r \cdot v_a)\, n/d \quad r = (v_r \cdot v_b)\, n/d$$

$$b = (v_u \cdot v_a)\, n/d \quad t = (v_u \cdot v_c)\, n/d$$

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

Projection matrix (P′) for each screen:

$$P' = PM^T T$$

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);
```

$$P' = \boxed{P} M^T T$$

Projection Matrix for CAVE Screens - M^T

- Review of the formula for M^T

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

- We already calculated v_r, v_u, and v_n
- So all we need to do is create a mat4 for M^T 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

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

Reminder:
  p_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

$$\boxed{P'} = PM^T T$$

Note:

P' 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:

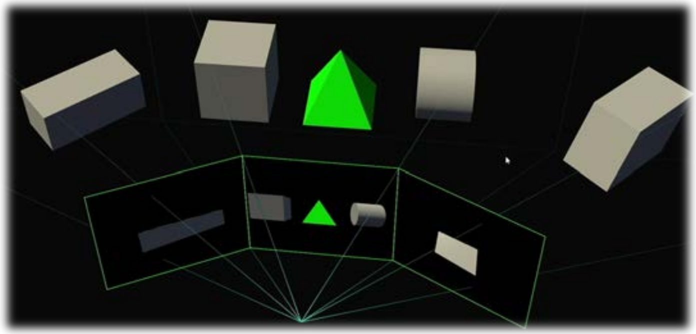
- Controller’s forward direction is perpendicular to your head forward direction
  - The image becomes mono
- Controller’s forward direction is in reverse direction
  - The image is inverted stereo

Resources

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

SMP

NVIDIA SMP (Simultaneous Multi-Projection)



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?v=p6NbyEmPalA