

5CM507 Graphics

Lecture 04 Viewing and Projection

Dr Youbing Zhao

October 1, 2025

Last Week



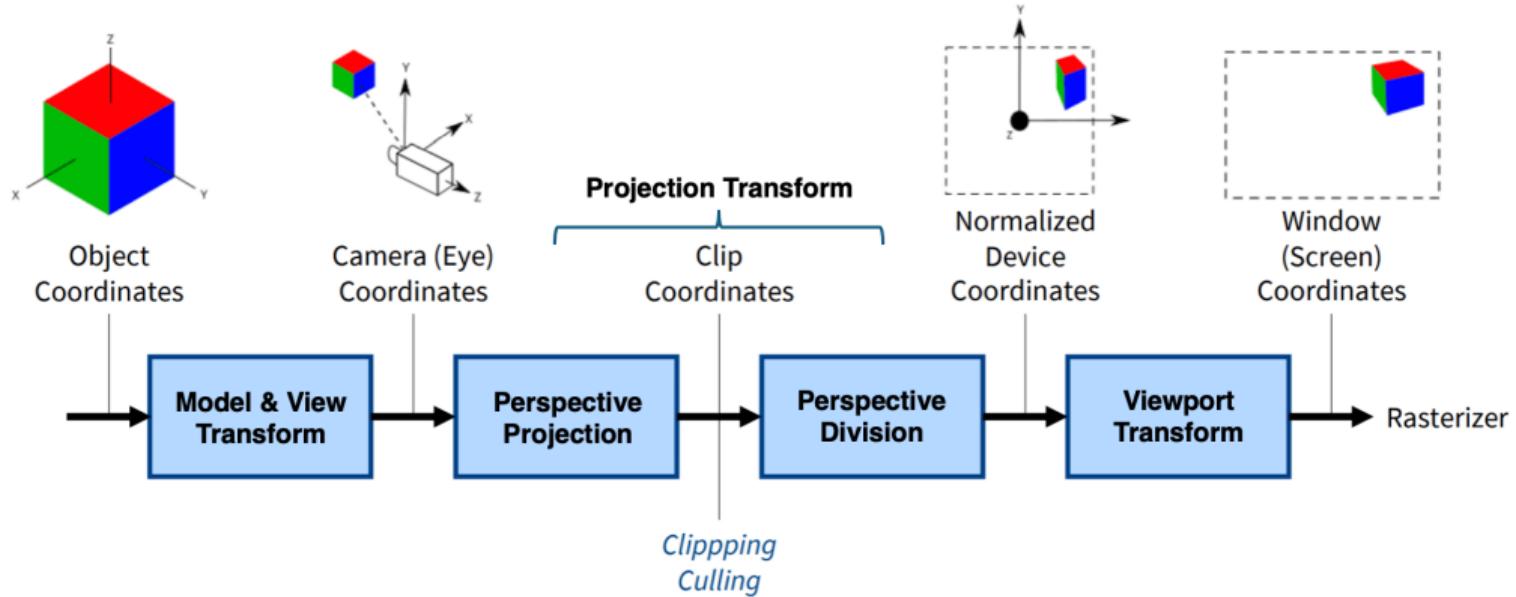
- ▶ Position a Model: Model Transform
- ▶ Build a scene: Hierarchical Transformation
- ▶ Transformation of frames
- ▶ Position and orient the Camera: View Transform
- ▶ The Model-View Matrix

Contents



- ▶ Camera control
- ▶ View volume and Normalised Device Coordinates (NDC)
- ▶ Orthographic Projection
- ▶ Perspective Projection

Modelling, Viewing and Projection chain



[Youngdo Lee]

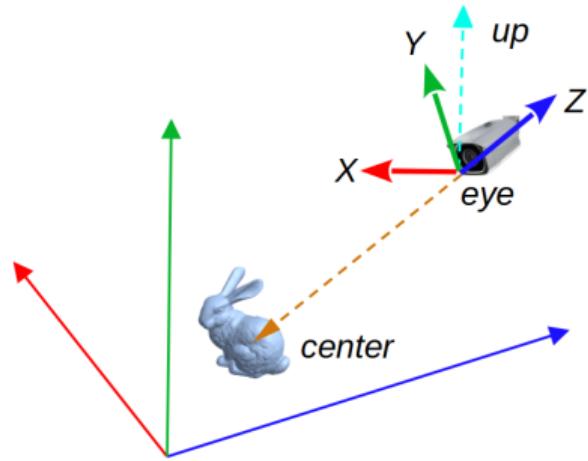
The View Matrix - the LookAt function

The LookAt function specifies the major axes of the camera space described in the world space

```
glm::dmat4 glm::lookAt(glm::dvec3 eye,  
    glm::dvec3 center, glm::dvec3 up);
```

The LookAt function defines the view matrix.

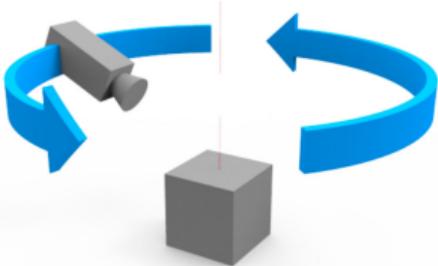
$$\mathbf{V} = \begin{bmatrix} {}^wX_c^x & {}^wX_c^y & {}^wX_c^z & -\mathbf{e} \cdot {}^w\vec{X}_c \\ {}^wY_c^x & {}^wY_c^y & {}^wY_c^z & -\mathbf{e} \cdot {}^w\vec{Y}_c \\ {}^wZ_c^x & {}^wZ_c^y & {}^wZ_c^z & -\mathbf{e} \cdot {}^w\vec{Z}_c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Basic Camera Control

Position and orientation

- ▶ Position and orient : `glm::lookAt()`
 - ▶ You can achieve the **bullet time effect** !
- ▶ Translation - change the position only
 - ▶ Easy way: use `glm::lookAt()` again
 - ▶ update the position
 - ▶ update the center based on the original orientation
 - ▶ But how about using translation matrices ?



Camera orbit



Camera dolly

Basic Camera Control



Translation

$$\mathbf{C} = \mathbf{PVM}$$

- ▶ Hint: a camera is move of $T(t_x, t_y, t_z)$ is equiv. to the scene/object's with the opposite move of $T(-t_x, -t_y, -t_z)$
- ▶ Move with regard to the world frame ${}^wT(t_x, t_y, t_z) \mathbf{C}' = \mathbf{PV} {}^wT(-t_x, -t_y, -t_z)\mathbf{M}$

$$\mathbf{v}' = \mathbf{v} {}^wT(-t_x, -t_y, -t_z) \begin{bmatrix} {}^wX_c^x & {}^wX_c^y & {}^wX_c^z & -(e + \vec{t}) \cdot {}^w\vec{X}_c \\ {}^wY_c^x & {}^wY_c^y & {}^wY_c^z & -(e + \vec{t}) \cdot {}^w\vec{Y}_c \\ {}^wZ_c^x & {}^wZ_c^y & {}^wZ_c^z & -(e + \vec{t}) \cdot {}^w\vec{Z}_c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- ▶ Move with regard to the camera frame ${}^cT(t_x, t_y, t_z) \mathbf{C}' = \mathbf{P} {}^cT(-t_x, -t_y, -t_z)\mathbf{V}\mathbf{M}$

$$\mathbf{v}' = \mathbf{v} {}^wT(-t_x, -t_y, -t_z) \begin{bmatrix} {}^wX_c^x & {}^wX_c^y & {}^wX_c^z & -e \cdot {}^w\vec{X}_c - t_x \\ {}^wY_c^x & {}^wY_c^y & {}^wY_c^z & -e \cdot {}^w\vec{Y}_c - t_y \\ {}^wZ_c^x & {}^wZ_c^y & {}^wZ_c^z & -e \cdot {}^w\vec{Z}_c - t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Basic Camera Control

Pan, Tilt, Zoom



$$\mathbf{C} = \mathbf{PVM}$$

- ▶ Pan : camera frame rotates θ_y about its Y axis
 - ▶ equiv. to the scene rotates $-\theta_y$ about the Y axis
- ▶ Tilt : camera frame rotates θ_x about its X axis
 - ▶ equiv. to the scene rotates $-\theta_x$ about the X axis
- ▶ Zoom : Field of View angle of projection

A camera pan followed by a tilt:

$$\mathbf{C}' = \mathbf{P} R_{tilt}(-\theta_x) R_{pan}(-\theta_y) \mathbf{V} \mathbf{M}$$

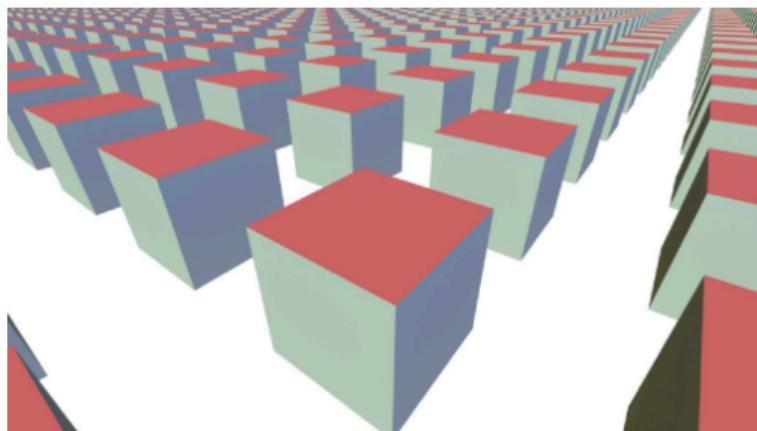


Projections

Perspective vs Orthographic Projections

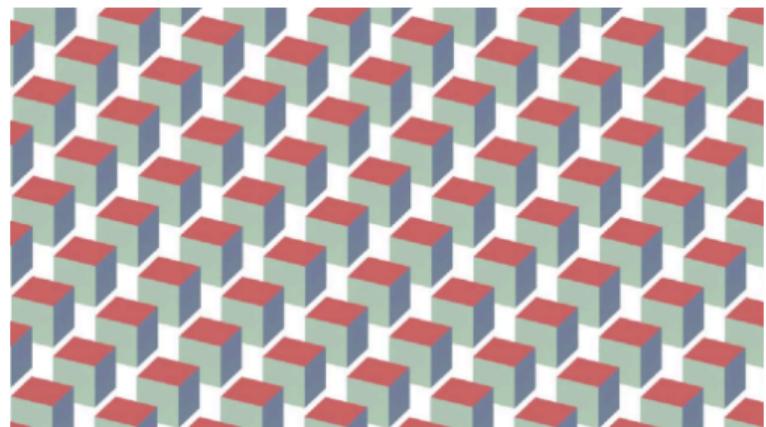
Perspective

- ▶ Does not preserve parallelism, proportions
- ▶ Provides a sense of depth
- ▶ 1 to 3 vanishing points

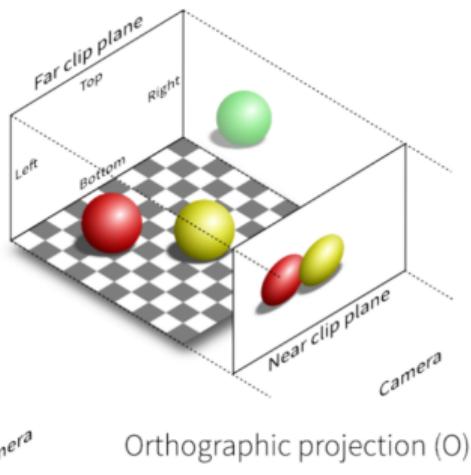
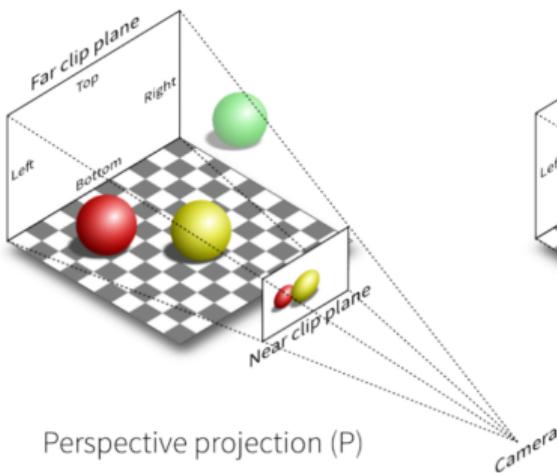


Orthographic

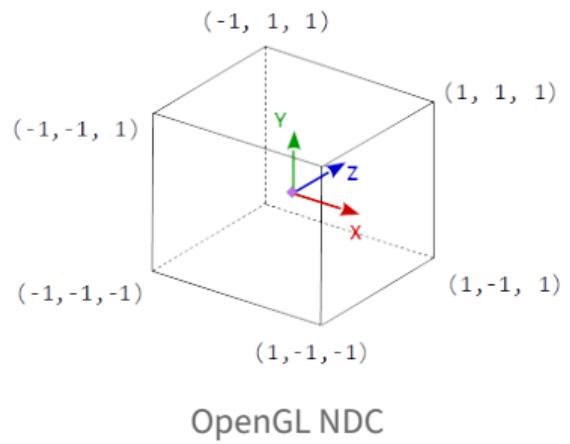
- ▶ A view preserving parallelism, proportions
- ▶ Lack a sense of depth



View volume and Normalised Device Coordinates (NDC)



Normalized Device Coordinates (NDC)

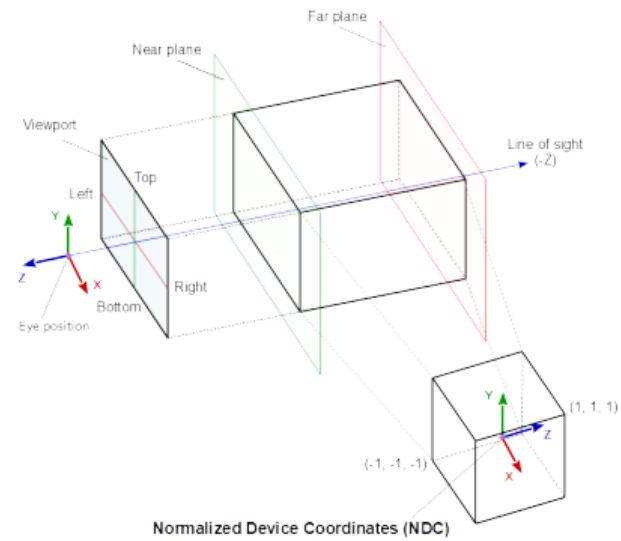
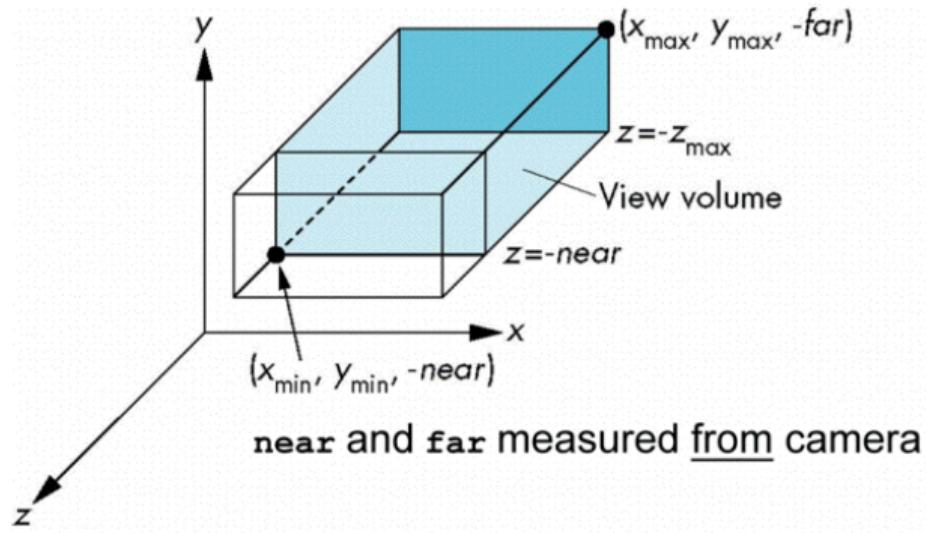


Orthographic Projection

Orthographic Projection

OpenGL orthographic projection function:

```
glm::mat4 glm::ortho(left, right, bottom, top, zNear, zFar);  
(left < right, bottom < top, 0 < near < far)
```



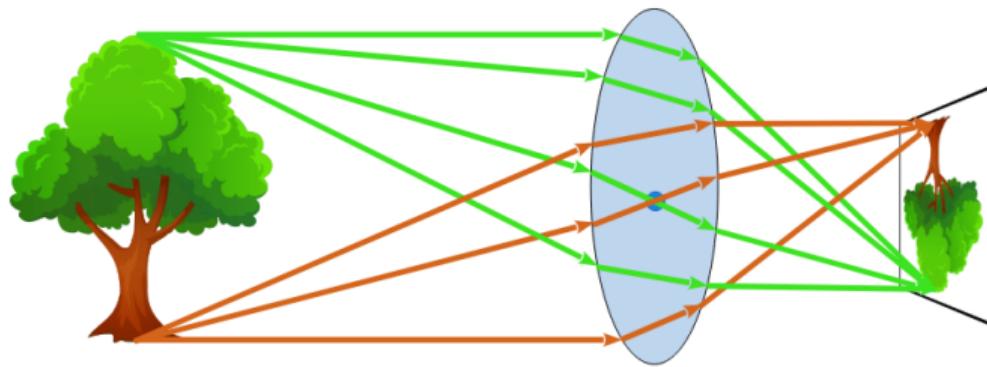
Orthogonal Projection Matrix

The orthogonal projection matrix is a translation to the centre of the view volume, following by scaling.

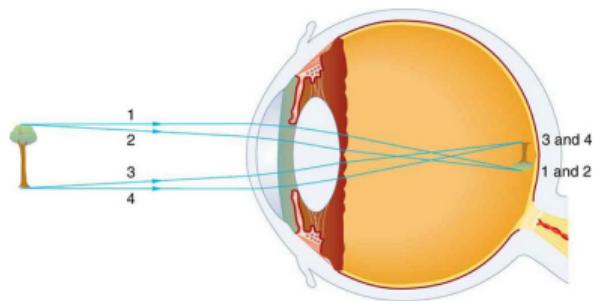
$$P = ST = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & -\frac{2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{right+left}{2} \\ 0 & 1 & 0 & -\frac{top+bottom}{2} \\ 0 & 0 & 1 & \frac{far+near}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Projection

A camera simulates an eye



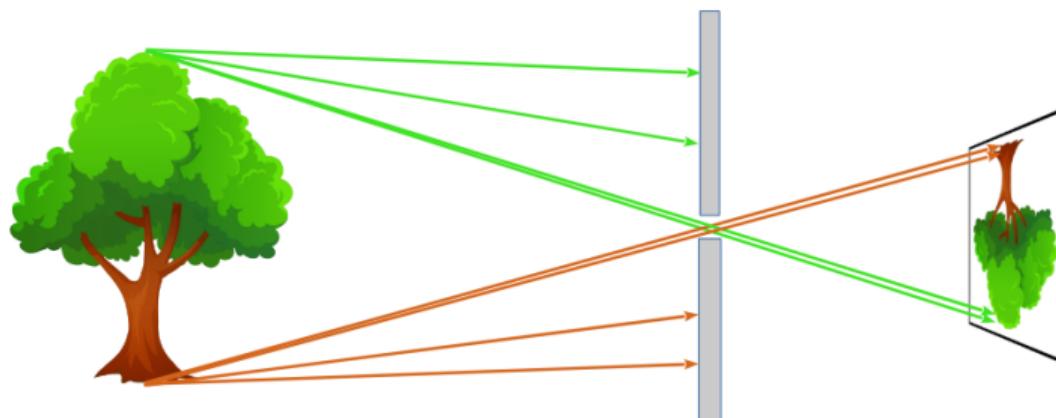
Imaging with lens



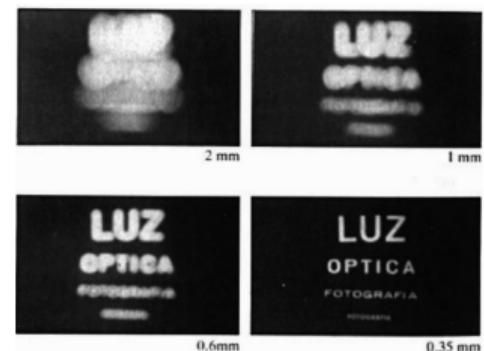
The human vision

The Pinhole Camera

- ▶ The pinhole effect : described by a great Chinese thinker Mo Di (Mozi) dating back to around 400BC.
- ▶ The pinhole camera model
 - ▶ computes the size of image projections
 - ▶ unable to simulate depth of field effects

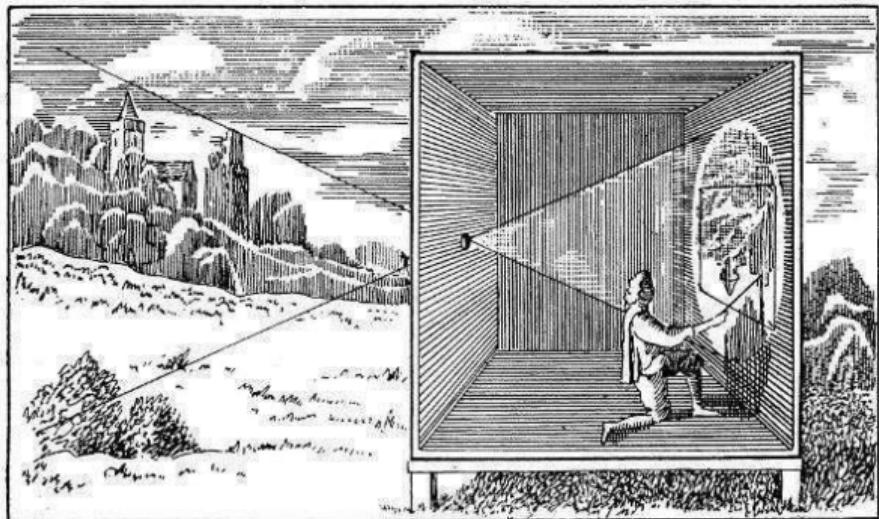


Imaging with a pinhole



Imaging with different pinhole sizes

Camera Obscura (dark chamber)



Engraving of a "portable" camera obscura in Athanasius Kircher's *Ars Magna Lucis Et Umbrae* (1645)

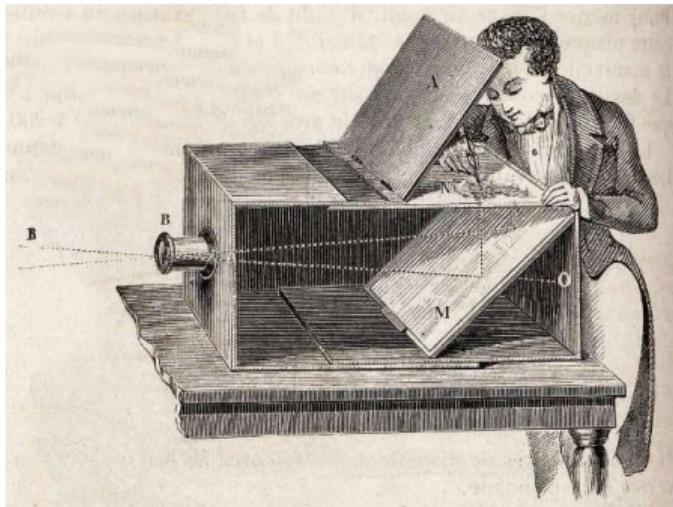
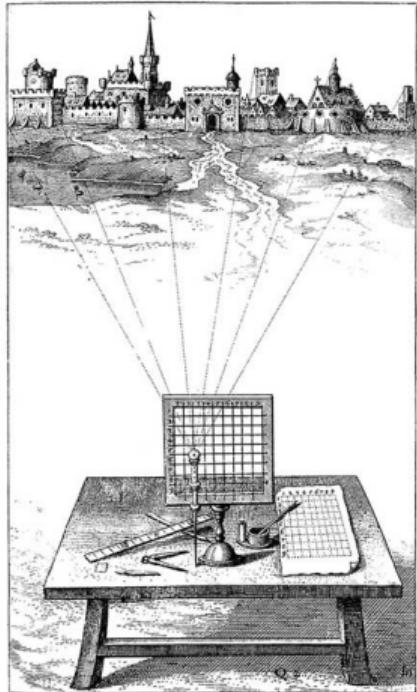
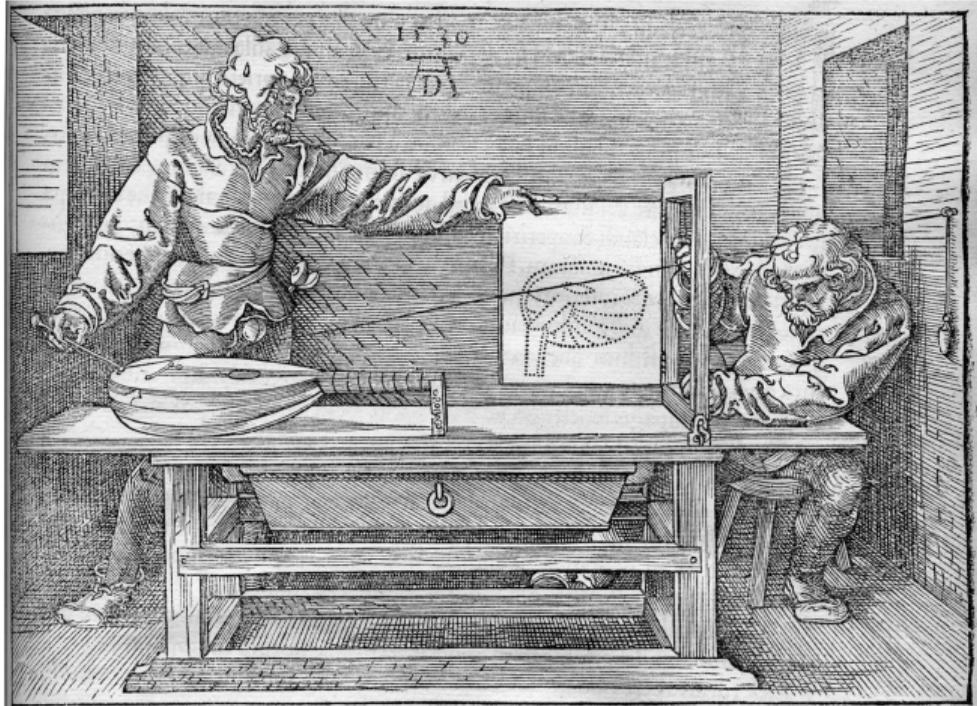


Illustration from Adolphe Ganot, *An Elementary Treatise on Physics*, 1882

Perspective Drawing Machines



Robert Fludd's sighting grid (1617)



Albrecht Dürer: "The draughtsman of the lute" (1525)

Perspective Drawing in Renaissance



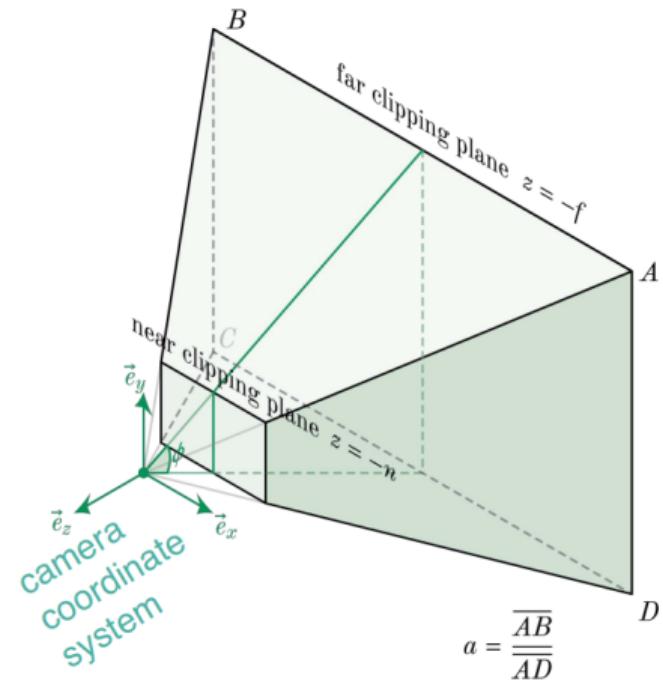
Annunciation, Leonardo da Vinci, c.1472-1476

Perspective View Frustum

Parameters

In graphics, a view frustum is the 3D region that is visible on the screen.

- ▶ Parameters for a viewing frustum:
 - ▶ field of view (FoV): angle ϕ
 - ▶ aspect ratio $a = \frac{\text{width}}{\text{height}}$
 - ▶ near clipping plan distance $n > 0$
 - ▶ far clipping plan distance $f > n$
- ▶ Camera zooming: adjusting the field of view angle ϕ
 - ▶ Wide angle zoom: wide FoV with a large ϕ
 - ▶ Telephoto zoom: narrow FoV with a small ϕ



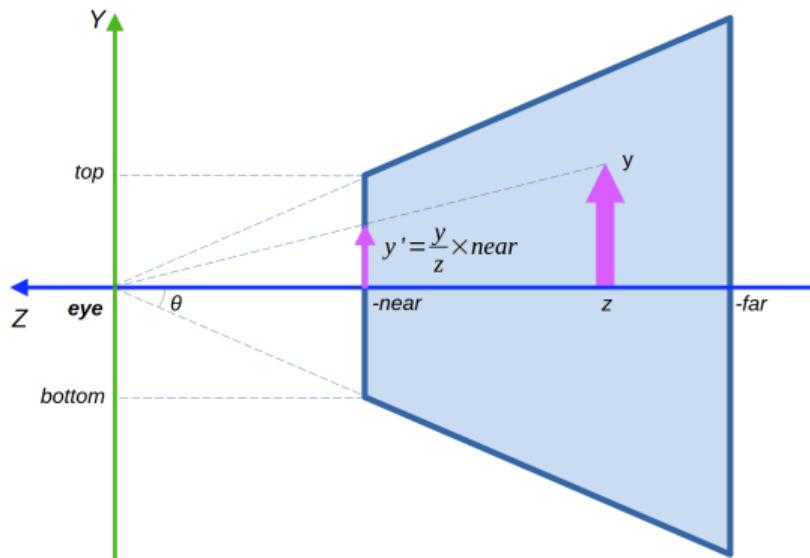
Perspective can be done simply as well ?

Let's have a try as the Renaissance draft man !

- ▶ Bring me my thread of simulated light.
- ▶ Bring me my target of sight.

$$\begin{cases} x' = \frac{x}{z}n \\ y' = \frac{y}{z}n \\ z' = \frac{z-n}{f-n} \end{cases}$$

- ▶ x and y are almost right
- ▶ this z' is OK for hidden surface removal
- ▶ but this does not preserve linearity
- ▶ We are projecting vertices
- ▶ Linearity is needed for clipping and interpolation
- ▶ We like to see a factor of $\frac{1}{z}$ in all of x', y', z'



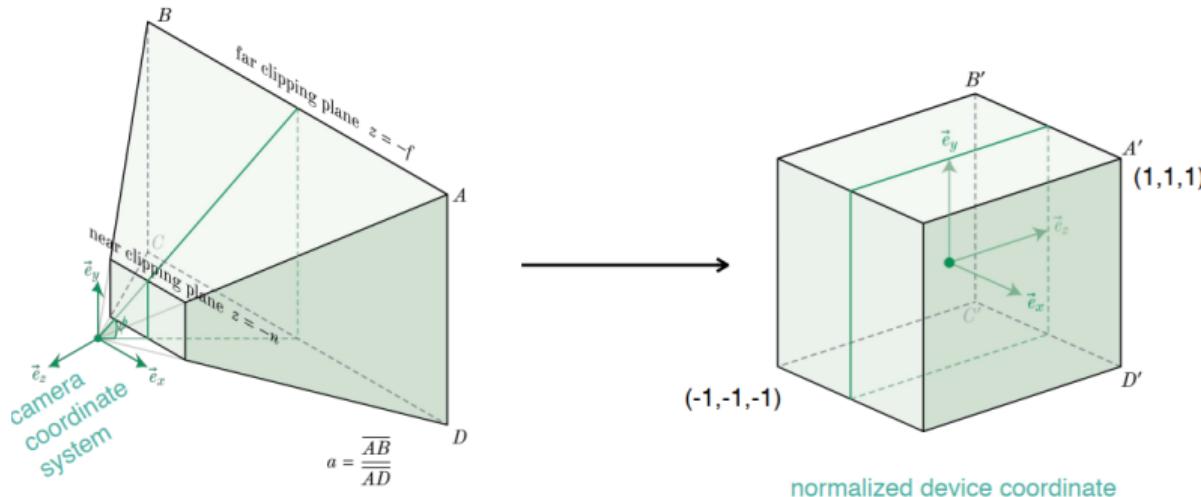
GLM functions for Perspective Projection

// General Viewing Volume

```
glm::mat4 glm::frustum(double left , double right , double bottom, double top, double zNear, double zFar);
```

// Symmetric Viewing Volume

```
glm::mat4 perspective(double fovy, double aspect, double zNear, double zFar);
```

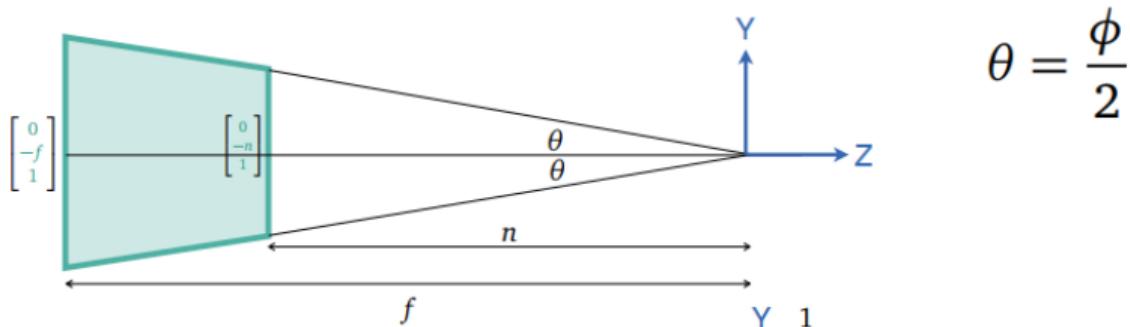


A projection matrix transforms a view frustum to a canonical cube

Symmetric Perspective Projection Matrix

A translation along Z, followed by a z-based scaling.

$$\mathbf{P}_{\text{sym}} \mathbf{v} = \begin{bmatrix} \frac{1}{a \tan \theta} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan \theta} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l}x \\ \frac{2n}{t-b}y \\ -\frac{f+n}{f-n}z - \frac{2fn}{f-n} \\ -z \end{bmatrix}$$



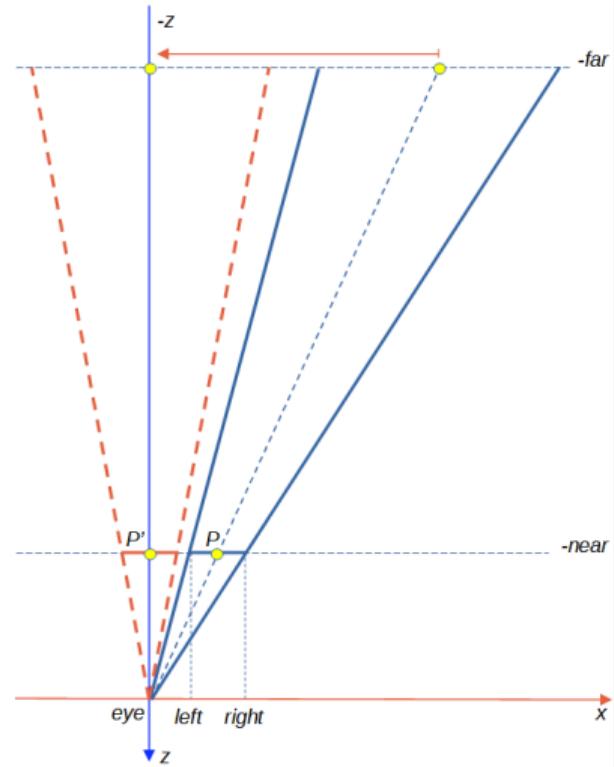
$$\theta = \frac{\phi}{2}$$

$$\tan \theta = \frac{top-bottom}{2near} \quad a \tan \theta = \frac{right-left}{top-bottom} \frac{top-bottom}{2near} = \frac{right-left}{2near}$$

The General Perspective Projection Matrix

- ▶ Oblique viewing volumes
 - ▶ Not common for single display games
 - ▶ Use a shear transform of X and Y to convert into a symmetric volume.

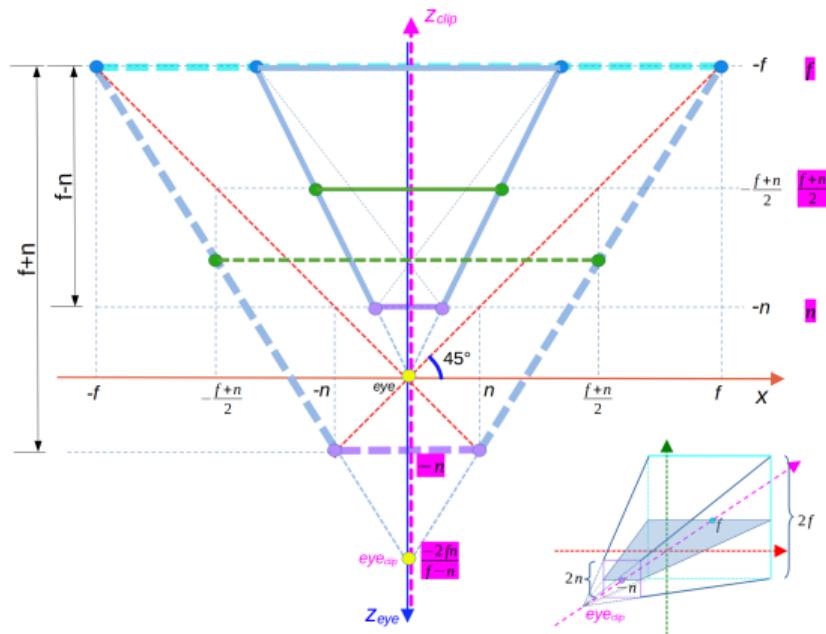
$$\mathbf{P} = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & \frac{r+l}{2n} & 0 \\ 0 & 1 & \frac{t+b}{2n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Projection step 1: the clip coordinates

The perspective matrix transforms a point $v(x, y, z, 1)$ to the clip space, before normalised by the weight $w_{clip} = -z$.

$$\begin{aligned} v_{clip} &= \mathbf{P}v = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{2n}{r-l}x + \frac{r+l}{r-l}z \\ \frac{2n}{t-b}y + \frac{t+b}{t-b}z \\ -\frac{f+n}{f-n}z - \frac{2fn}{f-n} \\ -z \end{bmatrix} = \begin{bmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{bmatrix} \end{aligned}$$



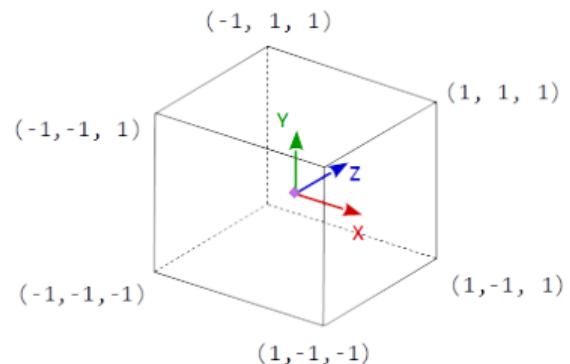
The clip volume: a frustum with a square near plane of $2n$ and a far plane of $2f$

Projection step 2: perspective divide (normalisation)

- ▶ Every cross section at $z = Z$ of the **view volume ranges** from $-Z$ to Z in both the x and y direction in the clip space, which simplifies clipping.
- ▶ Dividing x, y, z by the weight component $-z$ results in the normalised coordinate system (NCS).

$$v_{norm} = \begin{bmatrix} x_{norm} \\ y_{norm} \\ z_{norm} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{clip}/(-z) \\ y_{clip}/(-z) \\ z_{clip}/(-z) \\ (-z)/(-z) \end{bmatrix} = \begin{bmatrix} -\frac{2n}{r-l} \frac{x}{z} - \frac{r+l}{r-l} \\ -\frac{2n}{t-b} \frac{y}{z} - \frac{t+b}{t-b} \\ \frac{f+n}{f-n} + \frac{2fn}{(f-n)} \frac{1}{z} \\ 1 \end{bmatrix}$$

Normalized Device Coordinates (NDC)



OpenGL NDC

Distortions resulted from $\frac{1}{z}$

- ▶ Marginal distortion (large x, y offsets with a small $-z$ leading to large $\frac{x}{-z}$)
- ▶ Close-up distortion: (with a smaller n , $\frac{-z}{n} \rightarrow \infty$, $\frac{n}{z} \rightarrow 0$, projections converge to the vanishing point)
- ▶ Vertical distortion: buildings appears leaning if tilt the camera up (vanishing points caused by $\frac{1}{-z}$)



Vertical distortion



(a) A wide-angle photo with distortions on subjects' faces.

(b) Distortion-free photo by our method.

Marginal distortion [Google]



Wide angle distortion [Daniel]

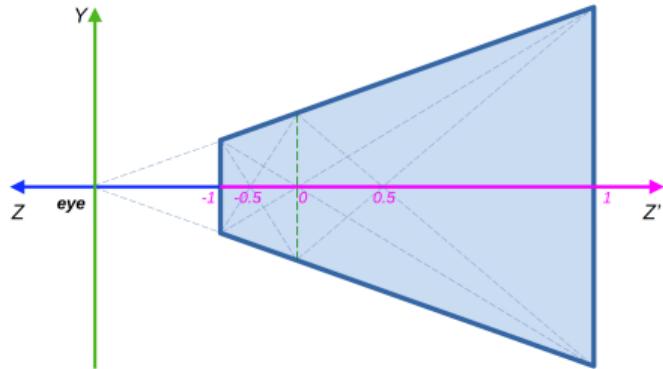
Transform of z is not uniform

$$z' = \frac{f+n}{f-n} + \frac{2fn}{f-n} \frac{1}{z}$$
 is not uniform

- ▶ Closer to the camera => greater numerical precision
- ▶ Most points are squeezed close to $z' = 1$
- ▶ May cause z-fighting when precision is lacking
- ▶ Points in the middle ranges are pushed much backwards away from the camera.

How to set near and far planes to make better use of the depth range

- ▶ Near plane: as far as possible
- ▶ Far plane: as near as possible



The z' quartiles before projection



Imagine how z' is squeezed [TCDD]

Summary



- ▶ Camera control
- ▶ Normalised Device Coordinates (NDC)
- ▶ Orthographic Projection Matrix
- ▶ Perspective Projection Matrices

Questions?

You are welcome to take the survey !