cs174A-dis1B-week3

October 18, 2019

1 CS-174A Discussion 1B, Week 3

@ Yunqi Guo

@ DODD 161 / Friday / 12:00pm-1:50pm

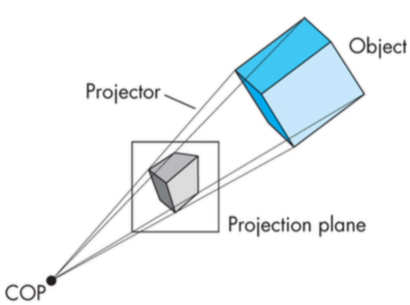
@ https://github.com/luckiday/cs174a-1b-2019f (Short link: https://bit.ly/32Zt3sg)

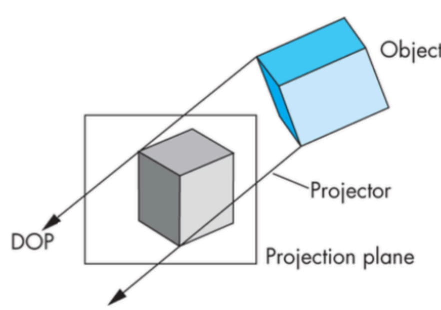
2 Outline

Viewing - Spaces - Model space - Objective/world space - Eye/camera space - Screen space - Projections - Parallel - Perspective

3 Objective

Our goal is to - Use transformations to project the vertices of objects onto the projection plane. - Specifically we will create transformations to go from object to camera to clip coordinates.

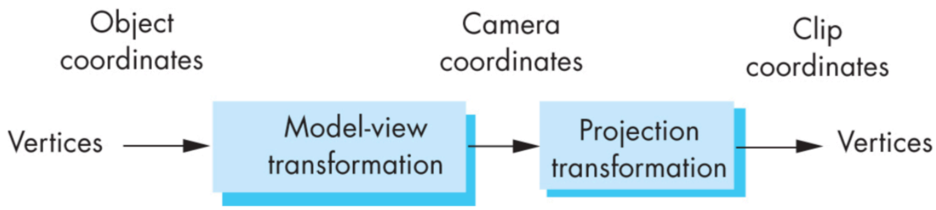




4 Model-view transformation

- Does not take us all the way to clip coordinates.
- We need a projection transformation for that.
- Model-view gets objects in front of the camera, potentially.

- A Projection defines which and how those objects will appear on the screen.

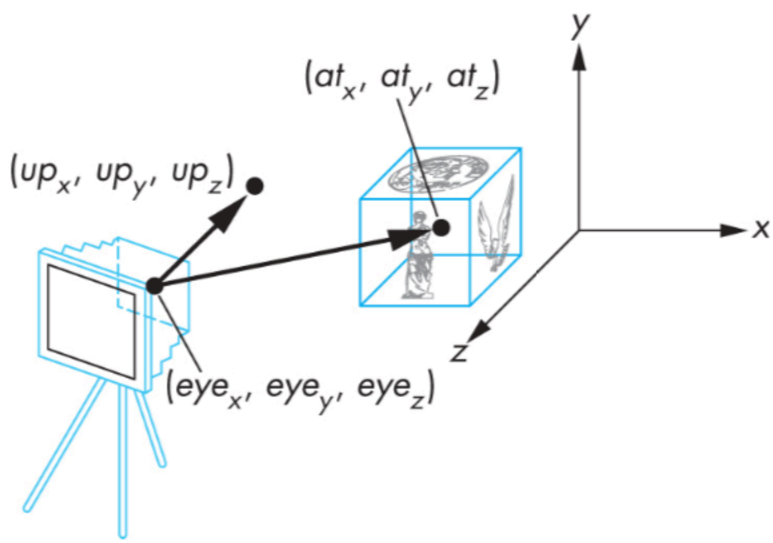


4.1 Positioning the (getting things in from of the) "camera"

- Recall that the default is "looking" down the –z axis at the origin (0,0,0).
  - This is equivalent to model-view set to the identity matrix.
- Remember, transformations are specified in reverse.
  - That means we specify the position of the camera first.

4.2 Look-at

We define three terms - A point describing the location of the eye. - A point the eye is looking at. - An up direction for the camera.



4.3 Look-at

- The at and eye points give us
  - the view-plane-normal or vpn
- The up vector is usually (0, 1, 0)
  - Or, (0, 1, 0, 0) in homogeneous coordinates!
- We then calculate the following

$$vpn = at - eye, \quad u = \frac{up \times n}{|up \times n|} n = \frac{vpn}{|vpn|}, \quad v = \frac{n \times u}{|n \times u|}$$

4.4 Look-at

- The at and eye points give us
  - the view-plane-normal or vpn
- The up vector is usually (0, 1, 0)
  - Or, (0, 1, 0, 0) in homogeneous coordinates!
- We then calculate the following

$$V = RT = \begin{pmatrix} u_x & u_y & u_z & -eye_x u_x - eye_y u_y - eye_z u_z \\ v_x & v_y & v_z & -eye_x v_x - eye_y v_y - eye_z v_z \\ n_x & n_y & n_z & -eye_x n_x - eye_y n_y - eye_z n_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5 Projection

- Perspective
- Parallel (orthographic)

5.1 Projections – Parallel (orthographic)

- Once in camera coordinates we need a projection transformation to get us to clip coordinates.
- The transformation matrix that gives us an orthographic projection is:

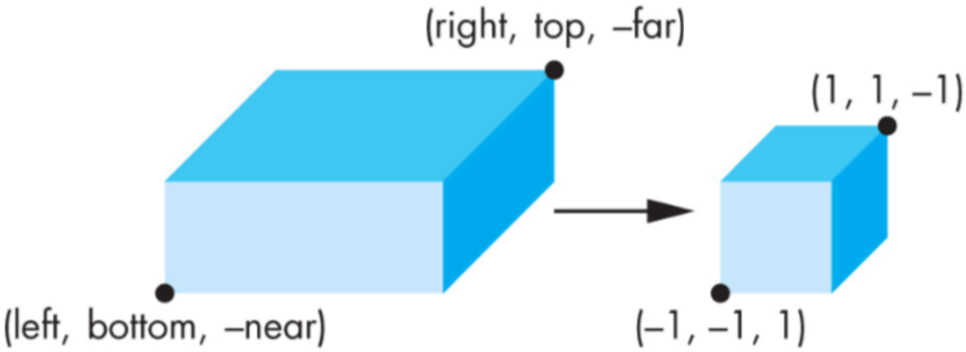
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.2 Projections – Parallel (orthographic)

- However, M is applied in the hardware after the vertex shader.
  - Which is in clip coordinates
- How do we "include" or "see" more of our scene?

5.3 Projections – Parallel (orthographic)

- We scale what we want to "include" to fit within the canonical view volume. i.e. (-1,1),(-1,1),(-1,1)
- Function in `tiny-graphics.js`: `js   orthographic(left, right, bottom, top, near, far)`



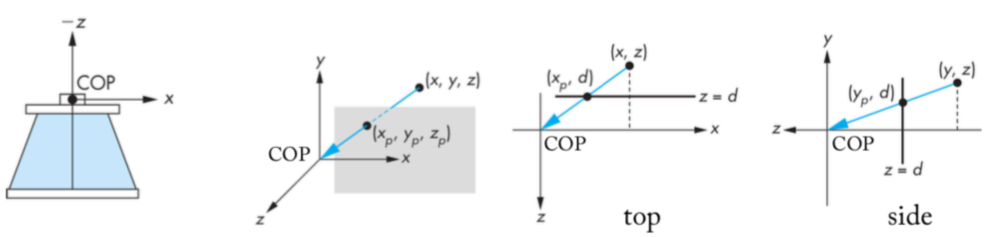
5.4 Projections – Parallel (orthographic)

$$N = \mathrm{ST} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & -\frac{n+2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.5 Projections – Perspective

- Basic symmetrical perspective projection
- The point (x, y, z) is projected through the projection plane to the eye point (or center of projection COP)
- We can compute the point of intersection with

$$x_p = \frac{x}{z/d}, y_p = \frac{y}{z/d}$$



5.6 Projections – Perspective

- The simple perspective projection matrix is

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix}$$

5.7 Projections – Perspective

- Uh oh, the homogeneous coordinate is no longer 1?

$$q = \begin{pmatrix} x \\ y \\ z \\ \frac{z}{d} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- We have to divide by the homogeneous coordinate to get back to 3D space.

$$q = \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x}{\frac{z}{d}} \\ \frac{y}{\frac{z}{d}} \\ d \\ 1 \end{pmatrix}$$

5.8 Projections – Perspective

Generalization!

$$N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

after perspective division, the point $(x, y, z, 1)$ goes to

$$x'' = -x/z, \quad y'' = -y/z, \quad z'' = -(a + b/z)$$

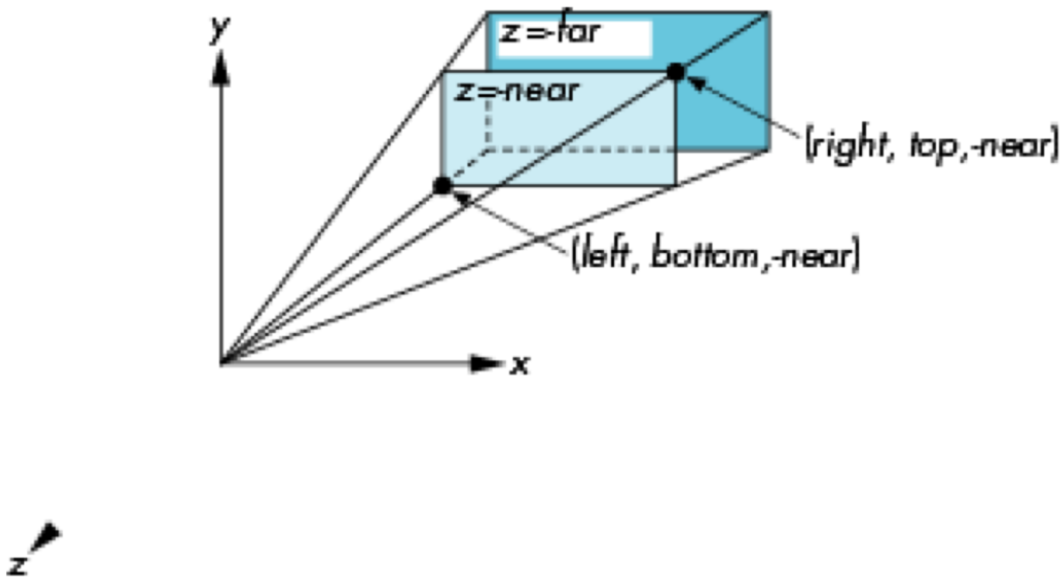
5.9 Projections – Perspective

- perspective projection matrix:

$$P = NSH = \begin{pmatrix} \frac{r}{n} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

5.10 Projections – Perspective

- WebGL Perspective js `frustum(left,right,bottom,top,near,far)`



5.11 Projections – Perspective

- Function in `tiny-graphics.js`: js `perspective(fov_y, aspect, near, far)`