Viewing

CMSC 435/634
Coordinate System / Space

- Origin + Axes
- Reference frame
- Convert by matrix
  - $\vec{p}_{\text{table}} = \text{TableFromPencil } \vec{p}_{\text{pencil}}$
  - $\vec{p}_{\text{room}} = \text{RoomFromTable } \text{TableFromPencil } \vec{p}_{\text{pencil}}$
  - $\vec{p}_{\text{room}} = \text{RoomFromPencil } \vec{p}_{\text{pencil}}$
Spaces

- **Object / Model**
  - Logical coordinates for modeling
  - May have several more levels

- **World**
  - Common coordinates for everything

- **View / Camera / Eye**
  - eye/camera at (0,0,0), looking down Z (or -Z) axis
  - planes: left, right, top, bottom, near/hither, far/yon

- **Normalized Device Coordinates (NDC) / Clip**
  - Visible portion of scene from (-1,-1,-1) to (1,1,1)

- **Raster / Pixel / Viewport**
  - 0,0 to x-resolution, y-resolution

- **Device / Screen**
  - May translate to fit actual screen
Nesting

- Room
  - Desk
    - Student
    - Book
    - Notebook
  - Desk
    - Student
    - Notebook
- Table
  - Laptop
- Blackboard
  - Chalk
  - Chalk
  - Eraser
Matrix Stack

- Remember transformation, return to it later
- Push a copy, modify the copy, pop
- Keep matrix and update matrix and inverse
- Push and pop both
Model $\rightarrow$ World / Model $\rightarrow$ View

- Model $\rightarrow$ World
  - All shading and rendering in World space
  - Transform all objects and lights
- Ray tracing implicitly does World $\rightarrow$ Raster
- Model $\rightarrow$ View
  - Serves just as well for single view
World → View

- Also called Viewing or Camera transform
- LookAt
  - $\vec{from}, \vec{to}, \vec{up}$
  - $\begin{bmatrix} \vec{u} & \vec{v} & \vec{w} & \text{from} \end{bmatrix}$
- Roll / Pitch / Yaw
  - Translate to camera center, rotate around camera
  - $R_z \ R_x \ R_y \ T$
  - Can have gimbal lock
- Orbit
  - Rotate around object center, translate out
  - $T \ R_z \ R_x \ R_y$
  - Also can have gimbal lock
View → NDC

- Also called *Projection* transform
- Orthographic / Parallel
  - Translate & Scale to view volume
    \[
    \begin{bmatrix}
    \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\
    0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\
    0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\
    0 & 0 & 0 & 1
    \end{bmatrix}
    \]
- Perspective
  - More complicated...
NDC $\rightarrow$ Raster

- Also called *Viewport* transform
- $[-1, 1], [-1, 1], [-1, 1] \rightarrow [-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$
  - Translate to $[0, 2], [0, 2], [0, 2]$
  - Scale to $[0, n_x], [0, n_y], [0, n_z]$
  - Translate to $[-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$

\[
\begin{bmatrix}
\frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\
0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\
0 & 0 & \frac{n_z}{2} & \frac{n_z-1}{2} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Raster→Screen

- Usually just a translation
  - More complicated for tiled displays, domes, etc.
- Usually handled by windowing system
Perspective View Frustum

- Orthographic view volume is a rectangular volume
- Perspective is a truncated pyramid or frustum
Perspective Transform

- Ray tracing
  - Given screen \((s^x, s^y)\), parameterize all points \(\vec{p}\)

- Perspective Transform
  - Given \(\vec{p}\), find \((s^x, s^y)\)
  - Use similar triangles
  - \(s^y/d = p^y/p^z\) So \(s^y = d p^y / p^z\)
Homogeneous Equations

- Same degree for every term
- Introduce a new redundant variable

\[ aX + bY + c = 0 \]
- \[ X = x/w, \ Y = y/w \]
- \[ a x/w + b y/w + c = 0 \]
- \[ \rightarrow a x + b y + c w = 0 \]

\[ aX^2 + bXY + cY^2 + dX + eY + f = 0 \]
- \[ X = x/w, \ Y = y/w \]
- \[ a x^2/w^2 + b x y/w^2 + c y^2/w^2 + d x/w + e y/w + f = 0 \]
- \[ \rightarrow a x^2 + b x y + c y^2 + d x w + e y w + f w^2 = 0 \]
Homogeneous Coordinates

- Rather than \((x, y, z, 1)\), use \((x, y, z, w)\)
- Real 3D point is \((X, Y, Z) = (x/w, y/w, z/w)\)
- Can represent Perspective Transform as 4x4 matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{bmatrix}
\begin{bmatrix}
p^x \\
p^y \\
p^z \\
1
\end{bmatrix}
= \begin{bmatrix}
p^x \\
p^y \\
p^z \\
p^z/d
\end{bmatrix}
\rightarrow \begin{bmatrix}
d & p^x/p^z \\
p^y/p^z \\
d & p^y/p^z \\
d & p^z/d & d
\end{bmatrix}
\]
Homogeneous Depth

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{bmatrix}
\begin{bmatrix}
p^x \\
p^y \\
p^z \\
1
\end{bmatrix}
= 
\begin{bmatrix}
p^x \\
p^y \\
p^z \\
p^z/d
\end{bmatrix}
\rightarrow
\begin{bmatrix}
d \frac{p^x}{p^z} \\
d \frac{p^y}{p^z} \\
d \frac{p^z}{d} \\
d
\end{bmatrix}
\]

- Lose depth information
- Can't get \( d \frac{p^z}{p^z} = p^z \)
  - Plus \( x/z, y/z, z \) isn't linear
- Use Projective Geometry
Projective Geometry

- If $x, y, z$ lie on a plane, $x/z, y/z, 1/z$ also lie on a plane
- $1/z$ is strictly ordered: if $z_1 < z_2$, then $1/z_1 > 1/z_2$
- New matrix:

$$
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
p^x \\
p^y \\
p^z \\
1
\end{bmatrix} = \begin{bmatrix}
p^x \\
p^y \\
p^z \\
1
\end{bmatrix} \rightarrow \begin{bmatrix}
p^x/p^z \\
p^y/p^z \\
p^z \\
1/p^z
\end{bmatrix}
$$
Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

\[
\begin{bmatrix}
a & 0 & 0 & 0 \\
0 & a & 0 & 0 \\
0 & 0 & b & c \\
0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y \\
p_z \\
1
\end{bmatrix}
= \begin{bmatrix}
a p_x \\
ap_y \\
b p_z + c \\
-p_z
\end{bmatrix} \rightarrow \begin{bmatrix}
-a p_x/p_z \\
-a p_y/p_z \\
-b - c/p_z
\end{bmatrix}
\]

- \(a = \cotan(\text{fieldOfView}/2)\)
- Solve for \(n \rightarrow -1\) and \(f \rightarrow 1\)
  - \(b = (n + f)/(n - f)\)
  - \(c = (2nf)/(f - n)\)
On Field of View

- Given image dimensions, set distance
  - Camera image sensor and focal length
- Given field of view angle in square window
- Non-square aspect ratio
  - Given horizontal (or vertical) field of view
  - Given diagonal field of view
- Off-center projection
  - Tiled displays
  - Head tracking