Theory behind Geometrical Transform
What does OpenGL do?

- So the user specifies a lot of information
  - Eye
  - Center
  - Up
  - Near, far,
  - Left, right top, bottom, etc.
What does OpenGL do?

- What does a system programmer do with those numbers?
- Generate screen coordinates *correctly* and *efficiently*
  - Inside/outside test
  - Projection
- *Here comes the part which contains math which you may not like*
- But all you need to know is matrix operation
Arbitrary View Volume
Inside-Outside Test

- Intersection of
  - A plane and
  - A Line

\[
\begin{align*}
\text{plane} & : ax + by + cz + d = 0 \\
\begin{cases}
x_1 + t(x_2 - x_1) \\
y_1 + t(y_2 - y_1) \\
z_1 + t(z_2 - z_1)
\end{cases} \\
a[x_1 + t(x_2 - x_1)] + b[y_1 + t(y_2 - y_1)] + c[z_1 + t(z_2 - z_1)] + d = 0 \\
t = -\frac{ax_1 + by_1 + cz_1 + d}{a(x_2 - x_1) + b(y_2 - y_1) + c(z_2 - z_1)} \\
0 \leq t \leq 1
\end{align*}
\]

\((x_1, y_1, z_1), (x_2, y_2, z_2)\) : end points of line
Clipping in Canonical Volumes

\[ Y = -Z \]

\[ Z = -1 \]

\[ (A,B,C) \]

\[ Z = \text{Zmin} \]

Y-min line

\[ Y = Z \]

\[ Z = 1 \]

Y-min line

\[ Y = -1 \]

far plane = film

near plane

Y-max line

\[ Y = 1 \]

Y-max line

\[ Z = -1 \]
Clipping with 6-bit outcode

- Perspective
- Above $y > -z$
- Below $y < z$
- Right $x > -z$
- Left $x < z$
- Behind $z < -1$
- In front $z > z_{\text{min}}$

- Parallel
- Above $y > 1$
- Below $y < -1$
- Right $x > 1$
- Left $x < -1$
- Behind $z < -1$
- In front $z > 0$
Projection

- Again, an intersection of
  - A plane and
  - A Line
Canonical Volumes

Y = -Z
Z = -1

(A/C, B/C)

near plane

Y = 1

(A, B)

Y = Z

Z = Z_{min}

Y = Z

far plane = film

Z = -1

Computer Graphics
Problem

- Both clipping and projection can be done efficiently in a canonical volume
- But we do not have a canonical volume in general
- Solution: Normalization transform
  - A single matrix operation to bring objects in any arbitrary volume into a canonical volume
  - Cannot change what the user sees
Case Study: Normalization Transform

- A transformation to facilitate clipping

An arbitrary view volume: Expensive for clipping and projection
The canonical view volume:
Simple clipping (six-bit outcode)
Simple projection (x/z, y/z)
OpenGL Terminology

UP

EYE

CENTER

f

b

X

Y

Z

top

d bottom

right

left
**PHIGS Terminology**

- **VUP**
- **VPN**
- **VRP**

Axes:
- **X**
- **Y**
- **Z**

**Parameters**:
- $u_{\text{min}}$
- $u_{\text{max}}$
- $v_{\text{min}}$
- $v_{\text{max}}$
Sidebar: Homogeneous Coordinates

- Inconsistent representation for translation
- Cannot be concatenated
- Homogeneous coordinates
  - consistent representation for all three
  - can be concatenated & pre-computed

\[
(x, y, z) \rightarrow (wx, wy, wz, w), w \neq 0
\]
\[
(wx, wy, w) \rightarrow (wx/w, wy/w, wz/w)
\]
**Sidebar: Euler Angle Rotation**

\[
\begin{align*}
\begin{pmatrix}
    x' \\
y' \\
z' \\
1
\end{pmatrix}
&=egin{pmatrix}
    \cos \theta & -\sin \theta & 0 & 0 \\
    \sin \theta & \cos \theta & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix} \\
\begin{pmatrix}
    x' \\
y' \\
z' \\
1
\end{pmatrix}
&=egin{pmatrix}
    \sin \theta & 0 & \cos \theta & 0 \\
    0 & 1 & 0 & 0 \\
    \cos \theta & 0 & -\sin \theta & 0 \\
    0 & 0 & \cos \theta & \sin \theta
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix} \\
\begin{pmatrix}
    x' \\
y' \\
z' \\
1
\end{pmatrix}
&=egin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & \cos \theta & -\sin \theta & 0 \\
    0 & \sin \theta & \cos \theta & 0 \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\end{align*}
\]
Sidebar: Rotation Matrix

- An orthonormal matrix
  - Have orthogonal rows
  - Have orthogonal columns
  - Does not magnify or shrink size of vector (eigen value is 1)
**SideBar: Rotation**

- **From world to eye:**
  - Column vectors are the $x$ (1,0,0), $y$ (0,1,0), $z$ (0,0,1) of the world frame in the eye frame.

- **From eye to world:**
  - Row vectors are the $x$, $y$, $z$ of the eye frame in the world frame.

$$
\begin{align*}
R^i_j &= \begin{bmatrix}
    r_x & r_y & r_z \\
    | & | & |
\end{bmatrix} \\
-

R^i_j &= \begin{bmatrix}
    -r'_x & - \\
    -r'_y & - \\
    -r'_z & - \\
\end{bmatrix}
\end{align*}
$$
SideBar: Rotation

- From world to eye
  \[ \mathbf{P}^i = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix} \mathbf{P}^j = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix} (\mathbf{x}_i + \mathbf{y}_i + \mathbf{z}_i) \]
  \[ = x r_x + y r_y + z r_z \]

- From eye to world
  \[ \mathbf{P}^j = \begin{bmatrix} -r'_x & -r'_y & -r'_z \end{bmatrix} \mathbf{P}^i = \begin{bmatrix} -r'_x & -r'_y & -r'_z \end{bmatrix} (\mathbf{x}_i + \mathbf{y}_i + \mathbf{z}_i) = (x, y, z) \]

\[ \mathbf{R}_{\text{camera}} \]
\[ \mathbf{P}_{\text{eye}} = \mathbf{R}_{\text{world}} \mathbf{P}_{\text{world}} \]
Comparison

- **PHIGS**
  - PRP (projection reference point)
  - VUP (viewup)
  - VPN (view plane normal)
  - VRP (view reference point)
  - umax, umin, vmax, vmin
  - View plane
  - F: front clipping distance
  - B: back clipping distance

- **OpenGL**
  - EYE
  - UP
  - EYE-CENTER
  - (left, bottom, -near)
  - right, left, top, bottom
  - N/A (or back clipping)
  - F
  - B
Normalization Transform

Perspective - OpenGL

- **External parameters**
  - Translate EYE into origin
  - Rotate the EYE coordinate system such that
    - w (e-c) becomes z
    - u becomes x
    - v becomes y

- **Internal parameters**
  - Shear to have centerline of the view volume aligning with z
  - Scale into canonical truncated pyramid
Existing Rendering Pipeline

- **Graphics primitives**
  - Transform matrix

- **Modeling transform**
  - Eye, lookat, headup

- **Viewing transform**
  - Parallel or Perspective volume

- **Clipping**
  - Material, lights, surface color

- **Shading & texture**

- **Projection**

- **Images in Internal buffer**
  - Viewport transform

- **Images on screen**
  - Viewport location

**Computer Graphics**
Rendering Pipeline with Normalization Transform

- Graphics primitives
- Modeling transform
- Normalization transform
- Viewport transform
- Viewing transform
- Clipping
- Shading & texture
- Projection
- Images in Internal buffer
- Images on screen
- Viewport location

Transforms:
- Transform matrix
- Eye, lookat, headup
- Parallel or Perspective volume
- Material, lights, surface color

Computer Graphics
Changes

- Modeling + Viewing + Normalization get concatenated into ONE transform before applying to any primitives
- Confusion: normalization does not just push the eye frame back to origin and line up with world frame, it pushes objects away too
- Purpose: to make clipping and projection much more efficient
Viewing Normalization

- Line up (X-Y-Z) and (U-V-W)
- Initially, (U-V-W) are specified in (X-Y-Z) system (In fact, everything is specified in X-Y-Z system)
- Some point in time, want to specify things in (U-V-W) system, or U becomes (1,0,0), V becomes (0,1,0), W becomes (0,0,1)
- Translation (easy) + Rotation (hard)
Translate EYE into the origin

\[
T_1 = \begin{bmatrix}
1 & 0 & 0 & -EYE_x \\
0 & 1 & 0 & -EYE_y \\
0 & 0 & 1 & -EYE_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Viewing Normalization

- Three rotations
  - Rotate about Y
  - Rotate about X
  - Rotate about Z
**Viewing Normalization**

- Figuring out \([u, v, w]\) in \([x, y, z]\) system
- Applying a rotation to transform \([x, y, z]\) coordinates into \([u, v, w]\) coordinates

\[
\begin{align*}
w &= \frac{e - c}{|e - c|} \\
u &= \frac{u \times w}{|u \times w|} \\
v &= w \times u
\end{align*}
\]

\[
\begin{bmatrix}
u \\ v \\ w \\ 1
\end{bmatrix} =
\begin{bmatrix}
u_x & \nu_y & \nu_z & 0 \\ u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\ y \\ z \\ 1
\end{bmatrix}
\]
Rotate EYE coordinate to align w. world system
Shear

\[ \text{left} + \text{right} \]
\[ \frac{\text{top} + \text{bottom}}{2} \]
\[ \frac{\text{near}}{} \]

\[ a = \frac{1}{2} \frac{\text{left} + \text{right}}{\text{near}} \]
\[ b = \frac{1}{2} \frac{\text{top} + \text{bottom}}{\text{near}} \]

\[ \text{Computer Graphics} \]
Scale into canonical volume

- scale in x and y
- scale in z

\[ S_1 = \left( \frac{\text{near}}{\text{right} - \text{left}}, \frac{\text{near}}{\text{top} - \text{bottom}}, 1 \right) \]

\[ S_2 = \left( \frac{1}{\text{far}}, \frac{1}{\text{far}}, \frac{1}{\text{far}} \right) \]
Example

\[ EYE = (10,10,10) \]
\[ CENTER = (0,0,0) \]
\[ UP = (0,1,0) \]
\[ (right, left) = (20,0) \]
\[ (top, bottom) = (20,0) \]
\[ F = 1 \]
\[ B = 10 \]
• Translate EYE into the origin

\[
T_1 = \begin{bmatrix}
1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 & 0 \\
0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

❖ Rotate EYE to align with the world system

\[
w = \frac{e - c}{|e - c|} = \frac{(1,1,1)}{\sqrt{3}}
\]

\[
u = \frac{\text{UP} \times w}{|\text{UP} \times w|} = \frac{(0,1,0) \times (1,1,1)}{|(0,1,0) \times (1,1,1)|} = \frac{(1,0,-1)}{\sqrt{2}}
\]

\[
v = w \times u = \frac{1}{\sqrt{6}} (1,1,1) \times (1,0,-1) = \frac{1}{\sqrt{6}} (-1,2,-1)
\]

\[
R = \begin{bmatrix}
\frac{1}{\sqrt{2}} & 0 & -1 & 0 & 0 \\
\frac{1}{\sqrt{2}} & 2 & \frac{\sqrt{2}}{1} & 0 & 0 \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 & 0 \\
\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & 0 & 1
\end{bmatrix}
\]

Computer Graphics
• Shear

\[
SH = \begin{bmatrix}
1 & 0 & 10 & 0 \\
0 & 1 & 10 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

\[
a = \frac{2}{\text{near}} + \frac{10}{\text{right}} = 10, \quad b = \frac{2}{\text{near}} + \frac{10}{\text{top}} = 10
\]
• Scale into canonical volume

• scale in x and y

\[
S_1 = \begin{bmatrix}
\frac{1}{10} & 0 & 0 & 0 \\
0 & \frac{1}{10} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

• scale in z

\[
S_1 = \begin{bmatrix}
\frac{1}{10} & 0 & 0 & 0 \\
0 & \frac{1}{10} & 0 & 0 \\
0 & 0 & \frac{1}{10} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Normalization Transform
Parallel (orthographic) - OpenGL

- **External parameters**
  - Translate EYE into origin
    - *Even though eye is not really where the viewer is*
  - Rotate the EYE coordinate system such that
    - w (e-c) becomes z
    - u becomes x
    - v becomes y

- **Internal parameters**
  - *Translate* to have centerline of the view volume aligning with z, and near plane at z=0
  - Scale into canonical *rectangular piped*
Viewing Normalization

- Line up (X-Y-Z) and (U-V-W)
- Initially, (U-V-W) are specified in (X-Y-Z) system (In fact, everything is specified in X-Y-Z system)
- Some point in time, want to specify things in (U-V-W) system, or U becomes (1,0,0), V becomes (0,1,0), W becomes (0,0,1)
- Translation (easy) + Rotation (hard)
Translate EYE into the origin

\[ T_1 = \begin{bmatrix} 1 & 0 & 0 & -EYE_x \\ 0 & 1 & 0 & -EYE_y \\ 0 & 0 & 1 & -EYE_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Viewing Normalization

- Three rotations
  - Rotate about Y
  - Rotate about X
  - Rotate about Z
**Viewing Normalization**

- Figuring out \([u, v, w]\) in \([x, y, z]\) system
- Applying a rotation to transform \([x, y, z]\) coordinates into \([u, v, w]\) coordinates

\[
\begin{align*}
  w &= \frac{e - c}{|e - c|} \\
  u &= \frac{u \times p \times w}{|u \times p \times w|} \\
  v &= w \times u
\end{align*}
\]

\[
\begin{bmatrix}
  u_x & u_y & u_z & 0 \\
  v_x & v_y & v_z & 0 \\
  w_x & w_y & w_z & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

*Computer Graphics*
Rotate EYE coordinate to align w. world system
Translation

\[ T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \]

\[ a = -\frac{\text{left} + \text{right}}{2}, \quad b = -\frac{\text{top} + \text{bottom}}{2}, \quad c = \text{near} \]
Scale into canonical volume

- Scale in x, y, and z

\[ S = \left( \frac{1}{right - left}, \frac{1}{top - bottom}, \frac{1}{far - near} \right) \]
Example

\[ \text{EYE} = (10,10,10) \]
\[ \text{CENTER} = (0,0,0) \]
\[ \text{UP} = (0,1,0) \]
\[ (\text{right}, \text{left}) = (20,0) \]
\[ (\text{top}, \text{bottom}) = (20,0) \]
\[ F = 1 \]
\[ B = 10 \]
• Translate EYE into the origin

\[ T_1 = \begin{bmatrix}
1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 & 0 \\
0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \]

* Rotate EYE to align with the world system

\[
\begin{align*}
w &= \frac{e - c}{\sqrt{3}} = \frac{(1,1,1)}{\sqrt{3}} \\
u &= \frac{\text{UP} \times w}{|\text{UP} \times w|} = \frac{(0,1,0) \times (1,1,1)}{|(0,1,0) \times (1,1,1)|} = \frac{(1,0,-1)}{\sqrt{2}} \\
v &= w \times u = \frac{1}{\sqrt{6}} (1,1,1) \times (1,0,-1) = \frac{1}{\sqrt{6}} (-1,2,-1)
\end{align*}
\]

\[
R = \begin{bmatrix}
\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\
\frac{1}{\sqrt{6}} & 2 & -\frac{1}{\sqrt{6}} & 0 & 0 \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 & 0 \\
\frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{6} & 0 & 1
\end{bmatrix}
\]
• Translation

\[
SH = \begin{bmatrix}
1 & 0 & 0 & -10 \\
0 & 1 & 0 & -10 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

\[
a = \frac{\text{left} + \text{right}}{\text{near}} = 10, \quad b = \frac{\text{top} + \text{bottom}}{\text{near}} = 10
\]
• Scale into canonical volume

• scale in x, y, and z

\[
S_1 = \begin{bmatrix}
\frac{1}{10} & 0 & 0 & 0 \\
0 & \frac{1}{10} & 0 & 0 \\
0 & 0 & \frac{1}{9} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]