Where are the vanishing points?

- Parallel lines after perspective projection are still parallel lines if they are also parallel to the projection plane

Why?

If \( L(t) = A + c \cdot t \) is a line

\[
A = (A_x, A_y, A_z) \quad ; \quad c = (c_x, c_y, c_z)
\]

if the view point (eye) is at the origin and the projection plane is perpendicular to the \( z \) axis at \(-d\), then the perspective projection of \( L(t) \) is:

\[
L_p(t) = (d \frac{A_x + c_x t}{-A_z - c_z t}, d \frac{A_y + c_y t}{-A_z - c_z t})
\]

(*)

If \( L(t) \) is parallel to the projection plane \((c_z = 0)\) then

\[
L_p(t) = \frac{-d}{A_z} \left( A_x + c_x t \right) , \quad \left( A_y + c_y t \right)
\]

Slope of \( L_p(t) \) is \( c_y / c_x \).
Parallel lines after perspective projection are no longer parallel lines if they are not parallel to the projection plane.

Why?

If \( L(t) \) is not parallel to the projection plane \((c_z \neq 0)\) then from (*) we that that

\[
L_p(t) \longrightarrow -d \left( \frac{c_x}{c_z}, \frac{c_y}{c_z} \right) \quad \text{when} \quad t \rightarrow \infty
\]

Hence, any line with the same direction vector would converge to this (vanishing) point

\[-d \left( \frac{c_x}{c_z}, \frac{c_y}{c_z} \right).
\]

Principal vanishing point: vanishing point generated by lines parallel to one of the principal axes (at most three PVPs).

Two-point perspective projection is popular.
How to find vanishing points?

Construct a line parallel to $AB$ that passes thru the view point (eye). The intersection of this line with the projection plane is the vanishing point of $AB$.
7.3 Camera Model for Perspective View

- How to create a perspective view of a scene in OpenGL?
- How to control the camera’s position and orientation in OpenGL?

Conceptual model of 3D viewing:

- Modelview matrix $M_v M_m$
- Projection matrix $M_p$
- Clip
- Perspective division

- Viewport matrix $M_{vp}$
- Normalized device coordinates
- Window (Device) coordinates
Define Viewing Coordinate System:
(specification of a 3D view)
(Positioning and pointing the camera)

```c
glMatrixMode ( GL_MODELVIEW );
glLoadIdentity ( );
gluLookAt ( eye.x, eye.y, eye.z, look.x, look.y, look.z, up.x, up.y, up.z);
```

\[ \mathbf{n} = \text{EYE} - \text{LOOK} \]
\[ \mathbf{u} = \text{UP} \times \mathbf{n} \]
\[ \mathbf{v} = \mathbf{n} \times \mathbf{u} \]
Define the view volume:
(create a camera model)

```
glMatrixMode ( GL_PROJECTION );
glLoadIdentity ( );
gluPerspective ( viewAngle, aspectRatio, N, F );
```

$N > 0, F > 0$
7.4 Building Viewing Matrix

View Pipeline

Canonical View Volume

- Parallel: \( x = \pm 1 \), \( y = \pm 1 \), \( z = \pm 1 \)

- Perspective: \( x = z \), \( x = -z \), \( y = z \), \( y = -z \), \( z = -z_{\text{min}} \), \( z = -1 \)
Modelview Matrix ($M_v, M_m$):

Modeling part ($M_m$):
- embodies all the modeling transformations for the object

Viewing part ($M_v$):
- accounts for the WC to VC transformation set by the camera’s position and orientation

\[
M_v = \begin{bmatrix}
    u_x & u_y & u_z & d_x \\
    v_x & v_y & v_z & d_y \\
    n_x & n_y & n_z & d_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

where

\[
(d_x, d_y, d_z) = (-u \cdot \text{eye}, -v \cdot \text{eye}, -n \cdot \text{eye})
\]
Projection Matrix ($M_p$):

$$M_p = \text{scaling2}$$

* translation

* perspective transformation

* scaling1

* shearing

$$M_p = M_{s2} \ast M_t \ast M_{pt} \ast M_{s1} \ast M_{sh}$$
Shearing:

- shear so that the center of the window would coincide with (0, 0, \(-N\))

\[
M_{sh} = \begin{bmatrix}
1 & 0 & a & 0 \\
0 & 1 & b & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
a = \frac{r + l}{2N}
\]

\[
b = \frac{t + b}{2N}
\]
Scaling1:

- scale so the user defined truncated view volume would coincide with the canonical view volume for perspective projection

\[
M_{s1} = \begin{bmatrix}
1/w & 0 & 0 & 0 \\
0 & 1/h & 0 & 0 \\
0 & 0 & 1/F & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
w = F \tan(\theta/2) \cdot AR
\]

\[
h = F \tan(\theta/2)
\]

\[
AR = \text{aspect ratio}
\]
Perspective Transformation:

- convert CVV for perspective projection to a quasi-CVV for parallel projection

\[
M_{pt} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{F}{F-N} & \frac{N}{F-N} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]
Translation:

- translate center of the quasi-CVV to the origin (0,0,0)

\[
M_t = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Scaling2:

- scale z-direction by 2 to get the CVV for parallel projection

\[
M_{s2} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
\[ M_p = M_{s2} M_t M_{pt} M_{s1} M_{sh} \]

\[
= \frac{1}{F} \begin{bmatrix}
\frac{2N}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\
0 & \frac{2N}{t-b} & \frac{t+b}{t-b} & 0 \\
0 & 0 & \frac{F+N}{F-N} & \frac{2FN}{F-N} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]