Realtime 3D Computer Graphics
Virtual Reality
Viewing and projection

Classical and General Viewing
Transformation Pipeline

object → Modelview Matrix → Projection Matrix → Perspective Division → Viewport Transform → other calculations here

- material → color
- shade model (flat)
- polygon rendering mode
- polygon culling
- clipping

Viewing

- Process for “Seeing” a world
- Projection of a 3D world onto a 2D plane
- Synthetic Camera Model
  - Location of viewer and view plane
  - What can be seen (Culling and Clipping)
  - How relationships are maintained
    - Parallel Lines
    - Angles
    - Distances (Foreshortening)
    - Relation to Viewer
- Objects vs. Scenes:
  - Some viewing techniques better suited for viewing single objects rather than entire scenes
  - Viewing an object from the outside (external viewing)
    - Engineering, External Buildings, …
  - Viewing an object from within (internal viewing)
    - Internal Buildings, Games, …
Viewing and projection

- Map transformed world from 3D to 2D.
  - by appropriate projection matrices.
- Several projection types exist:
  - Perspective projection is of major interest for 3D-CG.
  - Parallel projection importance
    - Pipeline often separates projection normalization from 3D-2D projection for additional depth testing.
- Calculation of the perspective projection for VR displays:
  - Display often require off-axis projection.
    - COPs are constantly moving.
    - Projection calculation has to be performed for 2 eyes.
  - Image planes might not be flat.

Definitions

- **Projection**: a transformation that maps from a higher dimensional space to a lower dimensional space (e.g. 3D->2D)
- **Center of projection (COP)**: the position of the eye or camera with respect to which the projection is performed (also eye point, camera point, proj. reference point)
- **Direction of projection (DOP)**: the direction of an eye or camera assumed to be infinite far away.
- **Projection plane**: in a 3D->2D projection, the plane to which the projection is performed (also view plane)
- **Projectors**: lines from coordinate in original space to coordinate in projected space
Projections

- **Perspective**: Distance to COP is finite
- **Parallel**: Distance to COP is infinite

**Planar Geometric Projections**
- Projection onto a plane
- Projectors are straight lines
- Alternatives:
  - Some Cartographic Projections
  - Omnimax
Parallel Projections

- Orthographic: Direction of projection is orthogonal to the projection plane
  - Elevations: Projection plane is perpendicular to a principal axis
    - Front
    - Top (Plan)
    - Side
  - Axonometric: Projection plane is not orthogonal to a principal axis
    - Isometric: Direction of projection makes equal angles with each principal axis.

- Oblique: Direction of projection is not orthogonal to the projection plane; projection plane is normal to a principal axis
  - Cavalier: Direction of projection makes a 45° angle with the projection plane
  - Cabinet: Direction of projection makes a 63.4° angle with the projection plane

Orthographic Projections

- “Special case” of perspective projection
- Parallel projectors perpendicular to projection plane

- Multiview
  - Projection Plane Parallel to Principle Faces
  - Classical Drafting Views
  - Preserves both distance and angles
  - Suitable to Object Views, not scenes
  - (a): Front-Elevation
  - (b): Top or Plan-Elevation
  - (c): Side-Elevation
Axonometric Projections

- Projection plane can have any orientation to object
- Parallel lines preserved
- Angles are not preserved
- Foreshortening:
  - Length is shorter in image space than in object space
  - Uniform Foreshortening (Perspective projections: foreshortening is dependent on distance from object to COP)

Oblique Projections

- Parallel Projections not perpendicular to projection plane
- Oblique projection types:
  - Cavalier: 45-degree Angles from Projection Plane
  - Cabinet: Arctan(2) or 63.4-degree Angles from Projection Plane
Perspective Projections

- First discovered by Donatello, Brunelleschi, and DaVinci during Renaissance
- Parallel lines appear to converge to single point
- Foreshortening:
  - Objects closer to viewer look larger
  - Length is not preserved
  - Depends on distance from viewer

Vanishing Points

- Perspective Projection of any set of parallel line (not perpendicular to the projection plane) converge to a *vanishing point*
- Infinity of vanishing points
  - one of each set of parallel lines
- Axis Vanishing Points
  - Vanishing point of lines parallel to one of the three principal axes
  - There is one axis vanishing point for each axis cut by the projection plane
  - At most, 3 such points
  - Perspective Projections are categorized by number of axis vanishing points
### Perspective Foreshortening

![Diagram of perspective foreshortening](image)

**How tall should this bunny be?**

### Projection Transformations

**NOTE:** Throughout the following discussions, we assume a camera coordinate system with COP at the origin and DOP along the z axis (OpenGL-like). The concepts are the same for any arbitrary viewing configuration.
Orthogonal Projections

- DOP parallel to z-axis
- Looking down negative z
- Display Plane at z=0
- Special Case of Perspective Projection

\[
x_p = x, \quad y_p = y, \quad z_p = 0
\]

\[
P_{orth} = M_{orth} p = \begin{bmatrix} x_p \\ y_p \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]
**Perspective Projection: Points onto a Plane**

- Projection of world onto display plane involves a perspective transformation:
  - $p' = M_{\text{per}}p$

  ![Perspective Projection Diagram](image)

- Observe:
  - $(b): \frac{x}{z} = \frac{x_p}{d} \iff x_p = \frac{x}{z/d}$
  - $(c): \frac{y}{z} = \frac{y_p}{d} \iff y_p = \frac{y}{z/d}$

  - Results in non-uniform foreshortening, **not** affine, **not** reversible

**One-point perspective projection**

- Center of Projection on the positive z-axis
- Viewplane in the x-y plane.

  ![One-point Perspective Projection Diagram](image)

  - Geometry of similar triangles.
  - Top view:
    - $x_p = \frac{xd}{d+z} = \frac{x}{1+(z/d)}$
    - $y_p = \frac{yd}{d+z} = \frac{y}{1+(z/d)}$
    - $z_p = 0$

  - $P_{\text{per}} = M_{\text{per}}P = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} x/(1+(z/d)) \\ y/(1+(z/d)) \\ 0 \\ 1 \end{bmatrix}$
One-point perspective projection

- Center of Projection on the positive z-axis
- Viewplane in the x-y plane.

- Geometry of similar triangles.
  Top view:
  \[ P : [x, y, z, 1] \]
  \[ [0, 0, d, 1] \]

- Point \( p \) in 3D become lines through origin in 4D:
  \( \text{As long as } w \neq 0, \text{ we can recover original point} \)

\[
p_{\text{per}} = M_{\text{per}}p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x/(1+(z/d)) \\ y/(1+(z/d)) \\ 0 \\ 1 \end{bmatrix}
\]

One-point perspective projection

- Center of Projection at origin
- Viewplane not in the x-y plane.

\[
p_{\text{per}} = M_{\text{per}}p = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} x/(z/d) \\ y/(z/d) \\ d \\ 1 \end{bmatrix}
\]
One-point perspective projection

- Center of Projection at origin
- Viewplane not in the x-y plane.

\[
\begin{align*}
p_{per} &= M_{per}p = \\
&= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix} =
\begin{bmatrix}
x/(z/d) \\
y/(z/d) \\
1 \\
1
\end{bmatrix}
\end{align*}
\]

- Point p in 3D become lines through origin in 4D: \( p = \begin{bmatrix} wx & wy & wz & w \end{bmatrix} \)

\[ \Rightarrow \text{As long as } w \neq 0, \text{ we can recover original point} \]

3D Viewing Transformation Projection Matrixes

- Input 3D World Coordinates
- Output 3D Normalized Device Coordinates
  - (a.k.a. Window Coordinates)
Projection normalization

- Several issues are not addressed with the simple projection matrices we have developed:
  - 3D Clipping Efficiency in a Frustum Viewing Volume
  - Hidden Surface Efficiency
- Solution: Use Projection Normalization
  - Get rid of perspective and other problem projections!
  - Everything is easier in a canonical, orthogonal world!
  - Distort world until viewing volume in world fits into a parallel canonical view volume.
  - Find a transformation that distorts the vertices in a way that we can use a simple canonical projection.

Idea: Canonical View Volume

- Define Viewing Volume via Canonical View Volumes
- Plus: Easier Clipping
- Minus: Another Transformation

OpenGL's canonical view volume (other APIs may be different):
Projection Normalization

• Distort world until viewing volume in world fits into a parallel canonical view volume.
• Find a transformation that distorts the vertices in a way that we can use a simple canonical projection.

Camera Transformation and Projection Normalization for Orthogonal Views

• Camera Transformation in Orthogonal Views ($M_{\text{cam}}$)
  1. Convert World to Camera Coordinates
     • Camera at origin, looking in the –z direction
     • Display plane center along the z axis
  2. Combinations of translate, scale, and rotate transformations
     • Can be accomplished through camera location specification

• Projection Normalization for Orthographic Views ($M_{\text{ortho}}$)
  1. Translate along the z axis until the front clipping plane is at the origin
  2. Scale in all three dimensions until the viewing volume is in canonical form
Projection Normalization for Orthogonal Parallel Projections

Mapping a orthogonal view volume to a canonical view volume requires two affine transformations:

\[
T = \begin{pmatrix}
-\frac{2}{y_{\text{max}} + y_{\text{min}}} & -\frac{2}{z_{\text{max}} + z_{\text{min}}} & -\frac{2}{x_{\text{max}} + x_{\text{min}}} \\
\frac{2}{y_{\text{max}} + y_{\text{min}}} & \frac{2}{z_{\text{max}} + z_{\text{min}}} & \frac{2}{x_{\text{max}} + x_{\text{min}}}
\end{pmatrix}
\]

These can be concatenated to

\[
M = ST = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]

The camera is pointing in the negative z direction. All projectors are from infinity towards the origin. Hence M can be written as:

\[
M = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]

OpenGL note:
OpenGL offers just an interface for the orthogonal case using

\[\text{glOrtho}(\text{xmin, xmax, ymax, ymin, near, far})\]

Finally, the resulting matrix has to be post multiplied by a simple orthogonal parallel projection

\[
M = M_{\text{ortho}}ST = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
M = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]

\[
M = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]

\[
M = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]

\[
M = \begin{pmatrix}
\frac{2}{x_{\text{max}} - x_{\text{min}}} & 0 & 0 & -\frac{x_{\text{max}} + x_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \\
0 & \frac{2}{y_{\text{max}} - y_{\text{min}}} & 0 & -\frac{y_{\text{max}} + y_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \\
0 & 0 & \frac{2}{z_{\text{max}} - z_{\text{min}}} & -\frac{z_{\text{max}} + z_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\end{pmatrix}
\]