EECS 487: Interactive Computer Graphics

Lecture 13:

- Planar Geometric Projections
- Orthographic projection
- Perspective projection
- Projections in OpenGL

Planar Geometric Projections

Planar == project onto a plane (vs. planetarium, e.g.)

Geometric == projectors are straight lines
(vs. curved lines in cartography, e.g.)

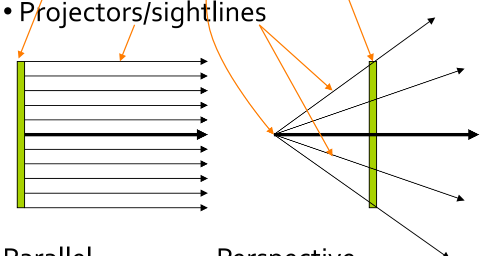
Projection == map from n to n-1 dimensions

Euclidean geometry describes shapes "as they are"
- properties of objects that are unchanged by rigid motions: lengths, angles, parallel lines

Projective geometry describes objects "as they appear"
- lengths, angles, parallel lines become "distorted" when we look at objects

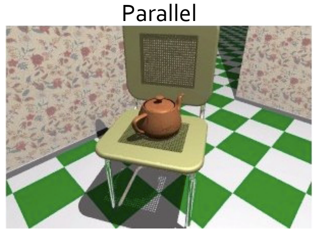
Projection System

Common elements:
- Center of Projection (COP) (for perspective projection)/ Direction of Projection (DOP) (for parallel projection, ≈COP at ∞)
- Projection/view/picture/image plane (PP)
- Projectors/sightlines



Parallel Perspective

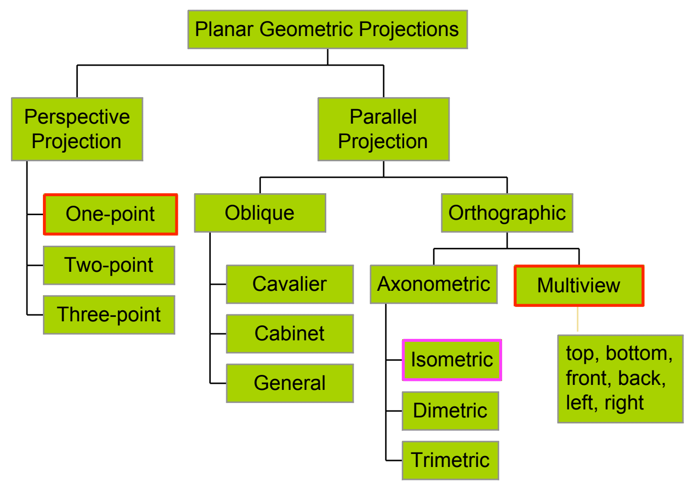
Parallel



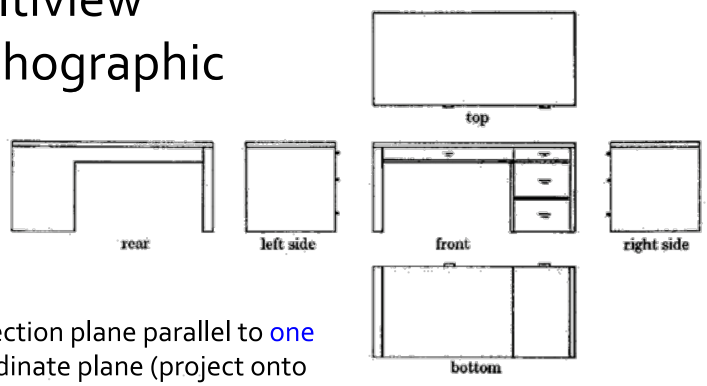
Perspective

Lozano-Perez01

Taxonomy of Planar Geometric Projections



Multiview Orthographic



- projection plane parallel to one coordinate plane (project onto plane by dropping coordinate perpendicular to plane)
- projection direction perpendicular to projection plane
- good for exact measurements (CAD, architecture)

- preserves ratios, but not angles (→ not visible)
- parallel lines remain parallel

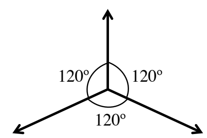
⇒ is considered an affine transform

James07

Axonometric Orthographic

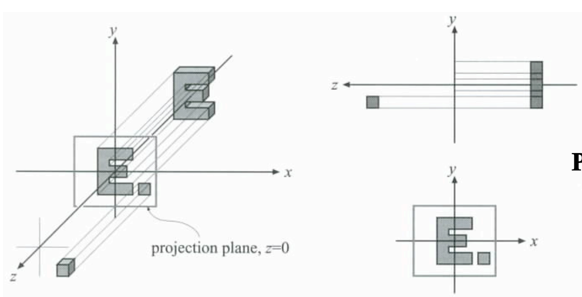
Axonometric:
- projection plane is not parallel to any coordinate plane
- projection direction perpendicular to projection plane

Isometric:
- preserves lengths along 3 principal axes
- principal axes make the same angle with each other (120º)



Age of Empires II
©Microsoft Corporation

Parallel Orthographic Projections



$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Are we done?

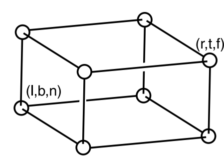
Yes, but we've lost depth (z) information, can't do:
- hidden surface removal
- lighting, etc.

Need to preserve z dimension!
⇒ map view volume to canonical view volume

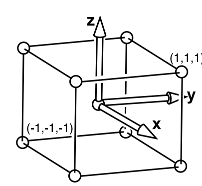
Akenine-Möller & Haines 02

Orthographic Projection

View volume defined by
left, right, bottom, top, near, and far planes:



Map it to cvv:

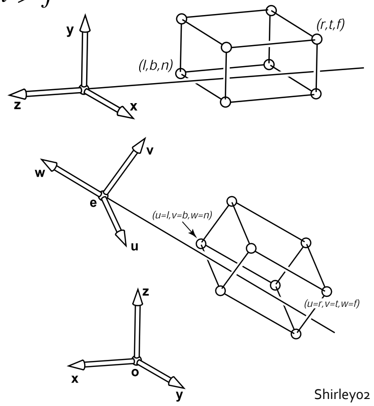


Shirley02

Orthographic Projection Setup

Simple case: view volume axis-aligned with world coordinate system

- the view volume is in negative z, $n > f$



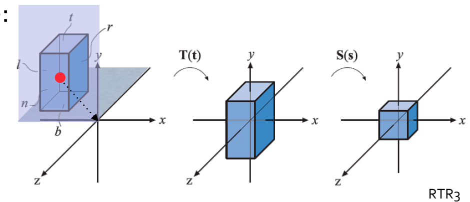
More generally, the view volume is not axis-aligned with world CS (it will always be axis-aligned with eye CS):

Shirley02

Orthographic Projection

From an arbitrary axis-aligned bounding box to canonical view volume

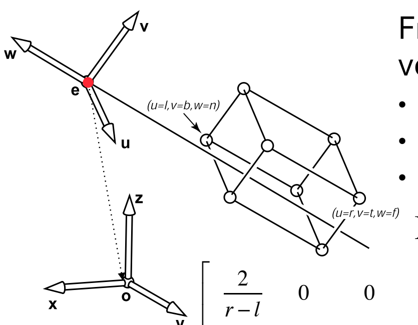
- translate and scale:



What would the $\mathbf{T}$ and $\mathbf{S}$ matrices be?

$$\mathbf{P}_o = \mathbf{ST} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic Projection and Viewing Transform



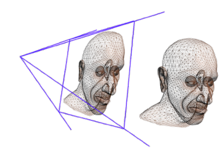
From an arbitrary viewing volume to canonical view volume

- translate eye to origin
- transform to eye coordinate system
- apply orthographic projection

$$\mathbf{M}_{world \to canonical} = \underbrace{\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_{eye \to canonical}} \underbrace{\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_{world \to eye}}$$

Shirley02

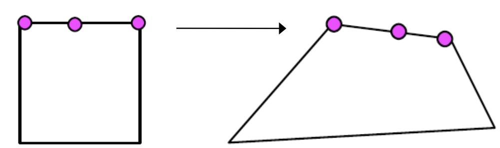
Perspective Projection



Objects appear smaller as distance from center of projection (eye of observer) increases (perspective foreshortening) $\Rightarrow$ looks more realistic (human eyes naturally see things in perspective)

Preserves:

- lines (collinearity)
- incidence ("lies on", intersects)
- cross ratio

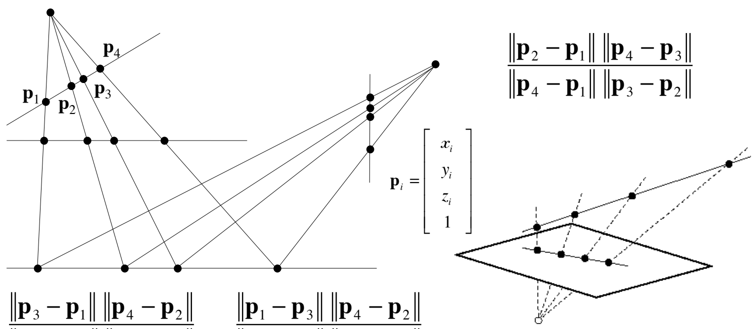


Does not always preserve parallel lines:

- lines parallel to projection plane remain parallel
- lines not parallel to projection plane converge to a single point on the horizon called the vanishing point (vp)

The Cross Ratio

For the 4 sets of 4 collinear points in the figure, the cross-ratio for corresponding points has the same value (can permute the point ordering)



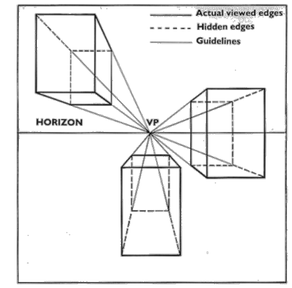
$$\frac{\|\mathbf{p}_2 - \mathbf{p}_1\| \, \|\mathbf{p}_4 - \mathbf{p}_3\|}{\|\mathbf{p}_4 - \mathbf{p}_1\| \, \|\mathbf{p}_3 - \mathbf{p}_2\|}$$

$$\frac{\|\mathbf{p}_3 - \mathbf{p}_1\| \, \|\mathbf{p}_4 - \mathbf{p}_2\|}{\|\mathbf{p}_3 - \mathbf{p}_2\| \, \|\mathbf{p}_4 - \mathbf{p}_1\|} \qquad \frac{\|\mathbf{p}_1 - \mathbf{p}_3\| \, \|\mathbf{p}_4 - \mathbf{p}_2\|}{\|\mathbf{p}_1 - \mathbf{p}_2\| \, \|\mathbf{p}_4 - \mathbf{p}_3\|}$$

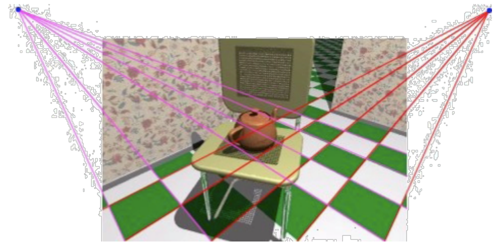
deMenthon
Seitz08

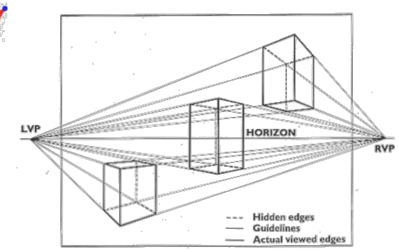
Classes of Perspective Projection

one-point: projection plane parallel to one coordinate plane (// to two coordinate axes, one coordinate axis cuts projection plane)



two-point: projection plane parallel to one coordinate axis (two coordinate axes cut projection plane)

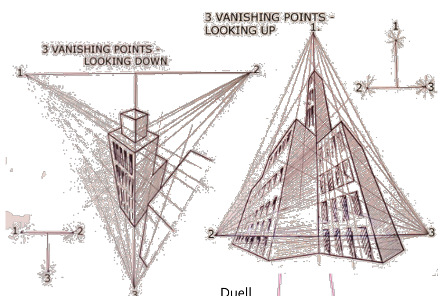




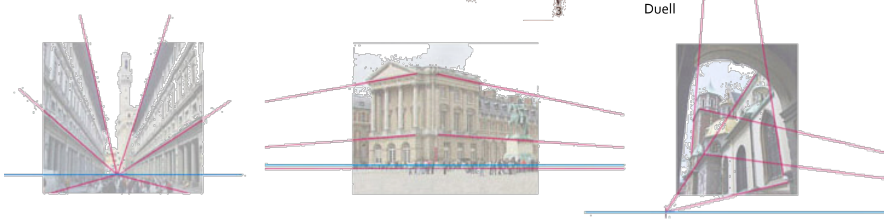
James07

Classes of Perspective Projection

three-point: projection plane not parallel to any coordinate axis (three coordinate axes cut projection plane)



Duell



Hulsey

James07

Projective Geometry in 2D

Consider lines and points in $\mathbf{P}$

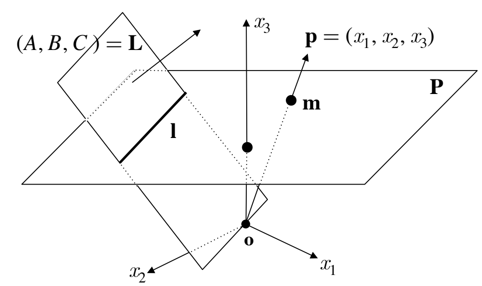
We extend to 3D to simplify dealing with infinity

- origin $\mathbf{o}$ out of $\mathbf{P}$, at a distance $= 1$ from $\mathbf{P}$

To each point $\mathbf{m}$ in $\mathbf{P}$ we can associate a single ray $\mathbf{p} = (x_1, x_2, x_3)$

To each line $\mathbf{l}$ in $\mathbf{P}$ we can associate a single plane (A, B, C)

- the equation of line $\mathbf{L}$ in projective geometry is $Ax_1 + Bx_2 + Cx_3 = 0$



deMenthon

Homogeneous Coordinates

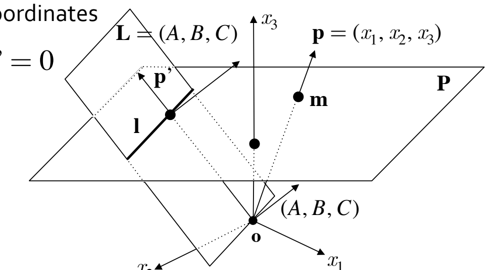
The ray $\mathbf{p} = (x_1, x_2, x_3)$ and $(\lambda x_1, \lambda x_2, \lambda x_3)$ are the same and are mapped to the same point $\mathbf{m}$ in $\mathbf{P}$

- $\mathbf{p}$ is the coordinate vector of $\mathbf{m}$, (x_1, x_2, x_3) its homogeneous coordinates

The planes (A, B, C) and $(\lambda A, \lambda B, \lambda C)$ are the same and are mapped to the same line $\mathbf{l}$ in $\mathbf{P}$

- $\mathbf{L}$ is the coordinate vector of $\mathbf{l}$, (A, B, C) its homogeneous coordinates

Point $\mathbf{p'}$ is on line $\mathbf{L}$ if $\mathbf{L} \cdot \mathbf{p'} = 0$



deMenthon

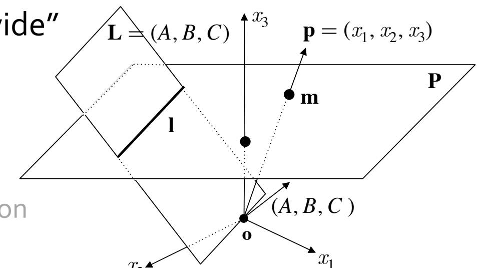
Perspective Divide

How do we "land" back from the projective world to the 2D Cartesian world of the plane?

- for point, consider the intersection of ray $\mathbf{p} = (\lambda x_1, \lambda x_2, \lambda x_3)$ with the plane $x_3 = 1 \Rightarrow \lambda = 1/x_3$, $\mathbf{m} = (x_1 / x_3, x_2 / x_3, 1)$
- for line, intersection of plane $Ax_1 + Bx_2 + Cx_3 = 0$ with the plane $x_3 = 1$ is line $\mathbf{l} = Ax_1 + Bx_2 + C = 0$

Called "perspective divide"

For the mathematically inclined, or studying computer vision: what's the geometric interpretation of $x_3 = 0$?



deMenthon

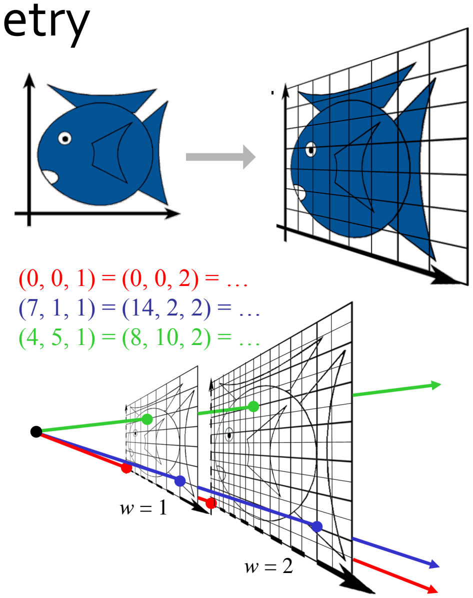
Projective Geometry

Two lines always meet at a single point, and two points always lie on a single line

Projective geometry does not differentiate between parallel and non-parallel lines

Points and lines are dual of each other

To return from homogeneous coordinates to Cartesian coordinates, divide by x_3 (w)



3D Projective Geometry

These concepts generalize naturally to 3D

Homogeneous coordinates

- projective 3D points have four coordinates: $\mathbf{p} = (x, y, z, w)$

Projective transformations

- represented by 4×4 matrices

Vanishing Points

What happens to two parallel lines that are not parallel to the projection plane?

The parametric equation for a line is:

$$\mathbf{l} = \mathbf{p} + t\mathbf{v} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

After perspective transform:

$$\begin{bmatrix} p'_x \\ p'_y \\ w \end{bmatrix} = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ (p_z + tv_z)/d \end{bmatrix} \xrightarrow{\text{after perspective divide}} \begin{bmatrix} \frac{p_x + tv_x}{p_z + tv_z} d \\ \frac{p_y + tv_y}{p_z + tv_z} d \\ 1 \end{bmatrix}$$

At the limit, with $t \rightarrow \infty$, we get a point! $[(v_x/v_z)d, (v_y/v_z)d, 1]^T$

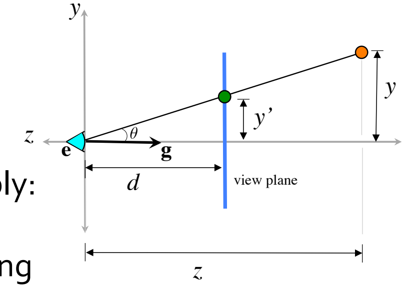
Each set of parallel lines intersect at a vanishing point

Perspective Projection

Given the coordinates of the orange point find the coordinates of the green point

$$\tan\theta = \frac{y'}{d} = \frac{y}{z}$$

$$y' = yd/z$$



Is perspective projection simply:
$(x, y, z, 1) \rightarrow (xd/z, yd/z, d, 1)$,
then map to screen by throwing away the z-coordinate: $(xd/z, yd/z, 1)$?

Perspective Projection Matrix

Projecting $(x, y, z, 1) \rightarrow (xd/z, yd/z, d, 1)$ and throwing away d does not preserve the depth information

Instead want $\mathbf{P}$ such that:

$$\mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x\frac{d}{z} \\ y\frac{d}{z} \\ z' \\ 1 \end{bmatrix}$$

perspective divide (first two rows)

preserve the relative depth information of each point (z')

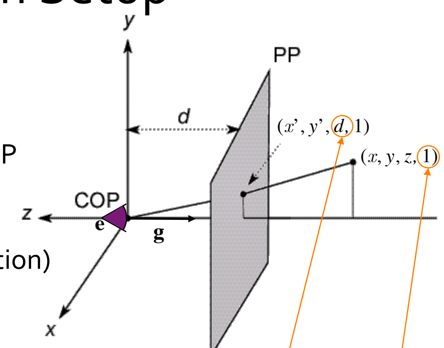
Just like orthographic projection, we need to map the view volume to a CVV instead of a 2D plane

Projection System Setup

The coordinate system

The eye (**e**)

- acts as the focal point and COP
- placed at the origin
- looking down (**g**) along the negative z-axis (axis of projection)



The screen

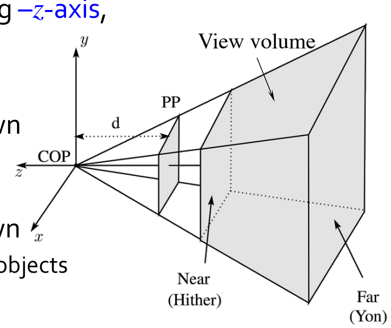
- lies in the projection plane
- $\perp$ to the z-axis, // to the x-y plane
- located at distance d from the eye
  - d is a.k.a. the focal length

What does it mean for $w=1$?
What does it mean for $w>1$?
What is the homogeneous coordinate (HC) when projecting from 3D to 2D?

Perspective Projection View Frustum

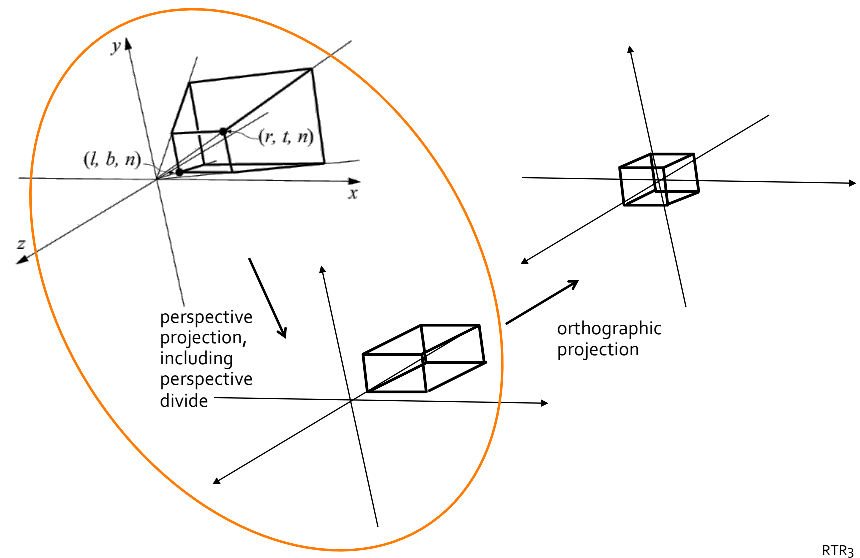
View volume (frustum: truncated pyramid):

- defined by (left, right, top, bottom, near, far) clipping planes
- near (n) and far (f) distances along $-z$-axis, both negative numbers, $n > f$
- nothing nearer than n will be drawn
  - → avoid numerical problems during rendering, such as divide by 0
- nothing further than f will be drawn
  - → avoid low depth precision for distant objects



To preserve relative depth information,
we must map the frustum to a CVV instead of a 2D plane

From Frustum to CVV



RTR3

Perspective Projection Matrix

Want projection matrix $\mathbf{P}$ such that: $\mathbf{P}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} xd/z \\ yd/z \\ z' \\ 1 \end{bmatrix}$

What should $\mathbf{P}$ be?

- we're projecting from 3D to 2D (not 4D to 3D), use the HC of the projected point to store its depth info (i.e., the "real" HC in 3D to 2D projection)
- first attempt:

$$\mathbf{Pp} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & z \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xd \\ yd \\ zd \\ z \end{bmatrix} \xrightarrow[\text{divide}]{\text{after perspective}} \begin{bmatrix} xd/z \\ yd/z \\ d \\ 1 \end{bmatrix}$$

Any problem?

Perspective Projection Matrix

Second attempt: for a more generic matrix, grab the depth info from the point itself:

$$\mathbf{Pp} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xd \\ yd \\ zd \\ z \end{bmatrix} \xrightarrow[\text{divide}]{\text{after perspective}} \begin{bmatrix} xd/z \\ yd/z \\ d \\ 1 \end{bmatrix}$$

Are we done?

- the projected x-, y-, and HC are correct already, but after perspective divide, all depths mapped to d!
- 3rd row of matrix must be tweaked to preserve relative depth info (z')

Perspective Projection Matrix

Let $d = n$

Want: $\mathbf{P}\underset{\text{Frustum}}{\begin{bmatrix}x\\y\\z\\1\end{bmatrix}} \to \underset{\text{Rectangular box}}{\begin{bmatrix}xn/z\\yn/z\\z'\\1\end{bmatrix}}$

The 1^{st} and 2^{nd} rows of P are correct already, for the 3^{rd} row (third attempt):

- the computation of z' does not rely on x and y, set the first two numbers of the row to 0
- we can use the remaining two numbers to compute z', let them be unknowns a and b for now:

$$\mathbf{Pp} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = \begin{bmatrix}xn\\yn\\az+b\\z\end{bmatrix} \xrightarrow[\text{divide}]{\text{after perspective}} \begin{bmatrix}xn/z\\yn/z\\a+b/z\\1\end{bmatrix}$$

Perspective Projection Matrix

For the 3^{rd} row of $\mathbf{P}$:

- want a and b such that:

$$\mathbf{P}\begin{bmatrix}x\\y\\n\\1\end{bmatrix} \to \begin{bmatrix}x\\y\\n\\1\end{bmatrix} \text{ and } \mathbf{P}\begin{bmatrix}x\\y\\f\\1\end{bmatrix} \to \begin{bmatrix}xn/f\\yn/f\\f\\1\end{bmatrix}$$

- or, for $z{=}n$, $a{+}b/z = n$ and for $z{=}f$, $a{+}b/z = f$

for $z = n:\ a + b/n = n,\ a = n - b/n$

for $z = f:\ a + b/f = f,$

substituting for a: $\left[(n - b/n) + b/f = f\right] nf$

$b(n-f) = (f-n)nf,\ b = -nf$

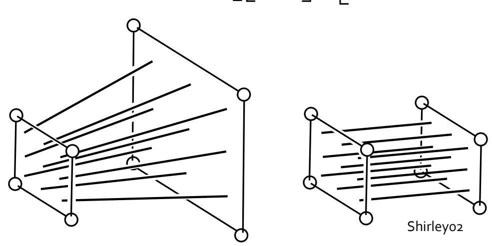
substituting for b: $a = n - (-nf)/n,\ a = n + f$

Perspective Divide

Then divide by the homogenous coordinate

⇒ squeezing the frustum into a rectangular box

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = \begin{bmatrix}nx\\ny\\(n+f)z-nf\\z\end{bmatrix} \xrightarrow[\text{divide}]{\text{perspective}} \begin{bmatrix}xn/z\\yn/z\\n+f-\frac{nf}{z}\\1\end{bmatrix}$$

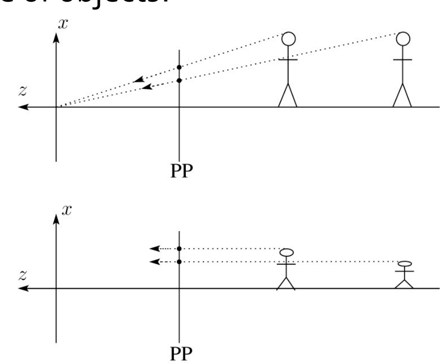


Shirley02

Note how n/z conveniently cancels the negative signs out

Perspective Foreshortening

What is the effect of perspective divide on the shape of objects?



⇒ After perspective divide, an object further away appears to be smaller than an equal-size object nearby

Curless08

From Frustum to CVV

Now reposition and scale the rectangular box

$$\mathbf{P}_p = \mathbf{STP} = \mathbf{P}_o\mathbf{P} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Assume viewing transform has been done, so after perspective divide (not shown) we're only dealing with axis-aligned viewing volume

$$= \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

See also http://www.songho.ca/opengl/gl_projectionmatrix.html

Losing Depth Precision

Recall that after perspective divide we have: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} nx/z \\ ny/z \\ n+f-\frac{fn}{z} \\ 1 \end{bmatrix}$

As a consequence of perspective foreshortening, z' is not linearly related to z:

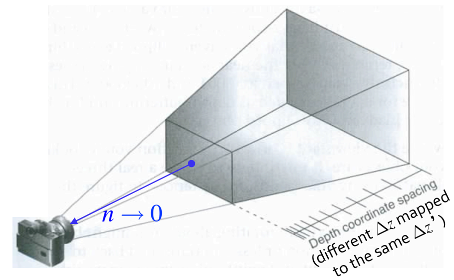
$$z' = n + f - \frac{nf}{z}$$

$$\Delta z' \approx \frac{nf\Delta z}{z^2};\ \Delta z \approx \frac{z^2}{fn}\Delta z'$$

max Δz is when $z = f$

At $z = f$, $\Delta z = \frac{f}{n}\Delta z'$, as $n \to 0$,

near the far plane (f), $\Delta z \to \infty$ but must be covered by the same $\Delta z'$ as smaller Δz that are closer to n



Redbook10

Losing Depth Precision

As a consequence of perspective foreshortening, z' is not linearly related to z: as z gets closer to f, the same amount of $\Delta z'$ must represent larger Δz

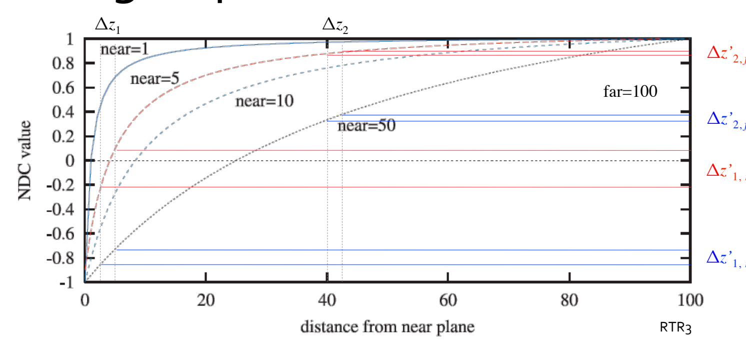
For example: let $n = 10, f = 90$,

for $z_1 = 10, z'_1 = 10$
for $z_2 = 11, z'_2 = 18.182$
$\Delta z_1 = 1$
$\Delta z'_1 = 8.182$

. . .

for $z_{k\text{-}1} = 89, z'_{k\text{-}1} = 89.888$
for $z_k = 90, z'_k = 90$
$\Delta z_{k\text{-}1} = 1$
$\Delta z'_{k\text{-}1} = 0.112$

Losing Depth Precision



RTR3

Implication of the non-linear mapping:

- information on the far plane loses precision
  $\Rightarrow$ z-buffer punch through or z-fighting
- distances closer to origin are exaggerated

Effect is ameliorated if n set further from origin

Akeley07

z-Buffer Quantization

z-values stored as non-negative integers

Integers are represented in b (=16 or 32) bits, giving a range of B ($= 2^b$) values $\{0, 1, 2, \ldots, B\text{-}1\}$

Floating point z'-values are discretized into integer bins: $\Delta z' = (f - n)/B$, so for example for $n = 10$, $f = 90$, both $z_1 = 89$, $z'_1 = 100-(900/89) = 89.888$ and $z_2 = 90$, $z'_2 = 100-(900/90) = 90$ are both discretized to $z' = 90$

Moral of the story: choose n as far away from origin as possible and f as near as possible (to reduce $\Delta z'$)