Projections

Steven Tanimoto

Adapted from materials by Brian Curless and Daniel Leventhal

CSE 457
Spring 2012

Reading

Required reading:

- Angel, 4.1 – 4.7 (Viewing: *Classical and Computer Viewing* through *Perspective-Projection Matrices*)

Further reading:

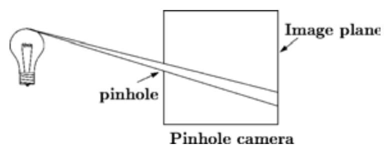
- Angel 4.8 – 4.10
- Foley, et al, Chapter 5.6 and Chapter 6
- David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, 2nd Ed., McGraw-Hill, New York, 1990, Chapter 2.
- I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, A characterization of ten hidden surface algorithms, *ACM Computing Surveys* 6(1): 1-55, March 1974.

Basic Camera

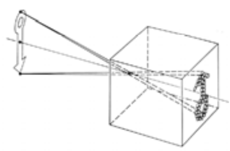
Office sized pinhole camera

The pinhole camera

The first camera - “camera obscura” - known to Aristotle.

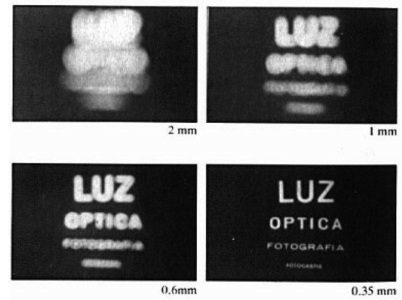


In 3D, we can visualize the blur induced by the pinhole (a.k.a., **aperture**):



Q: How would we reduce blur?

Shrinking the pinhole



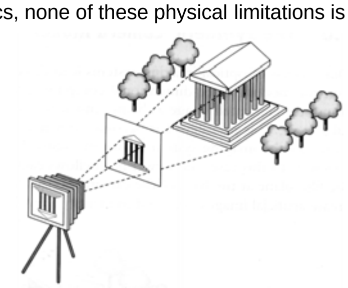
Q: What happens as we continue to shrink the aperture?

Shrinking the pinhole, cont’d

Imaging with the synthetic camera

In practice, pinhole cameras require long exposures, can suffer from diffraction effects, and give an inverted image.

In graphics, none of these physical limitations is a problem.



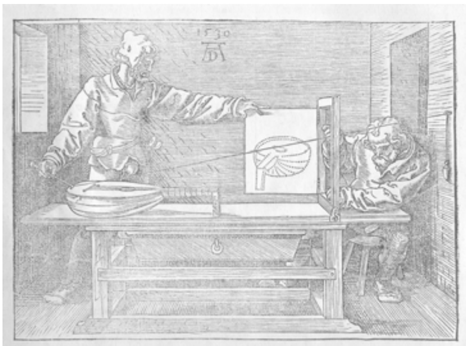
The image is rendered onto an **image plane** (usually in front of the camera).

Viewing rays emanate from the **center of projection** (COP) at the center of the pinhole.

The image of an object point *P* is at the intersection of the viewing ray through *P* and the image plane.

But is P visible? This the problem of **hidden surface removal** (a.k.a., **visible surface determination**).

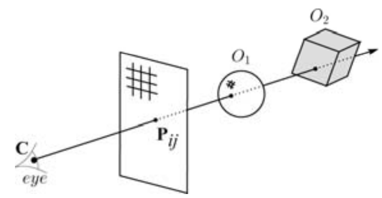
Ray casting by hand



[Durer. Institutiones Geometricae 1532]

Ray casting

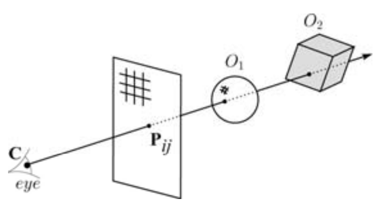
One way to simulate the pinhole camera and determine which point is visible at each pixel is **ray casting**.



<u>Idea:</u> For each pixel center $\boldsymbol{P}_{ij}$

- Send ray from eye point (COP), **C**, through $\boldsymbol{P}_{ij}$ into scene.
- Intersect ray with each object.
- Select nearest intersection.

Ray casting, cont.



<u>Implementation:</u>

- Might parameterize each ray:

  $\mathbf{r}(t) = \mathbf{C} + t\,(\mathbf{P}_{ij} - \mathbf{C})$

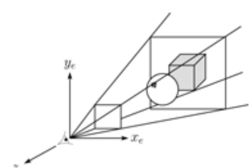
  where $t > 0$.

- Each object O_k returns $t_k > 0$ such that first intersection with O_k occurs at $\mathbf{r}(t_k)$.

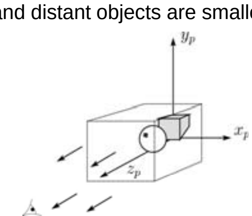
Q: Given the set $\{t_k\}$ what is the first intersection point?

Warping space

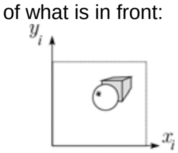
A very different approach is to take the imaging setup:



then warp all of space so that all the rays are parallel (and distant objects are smaller than closer objects):



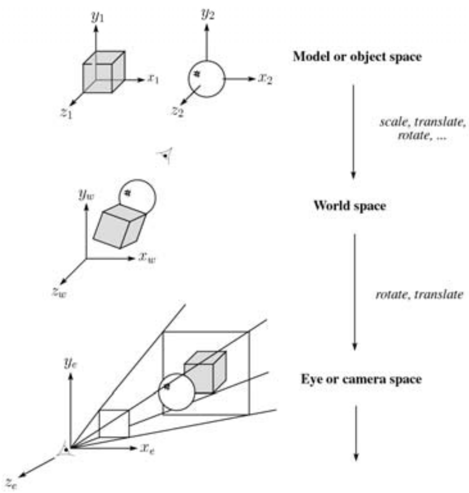
and then just draw everything onto the image plane, keeping track of what is in front:



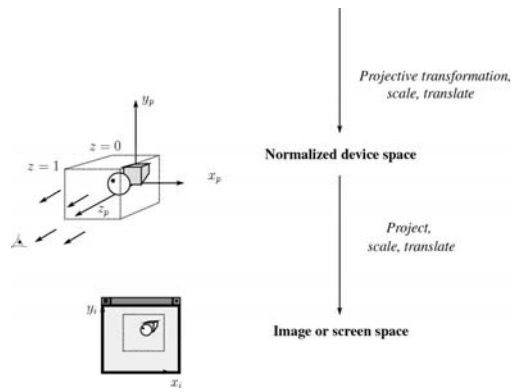
3D Geometry Pipeline

Graphics hardware follows the “warping space” approach.

Before being turned into pixels, a piece of geometry goes through a number of transformations...

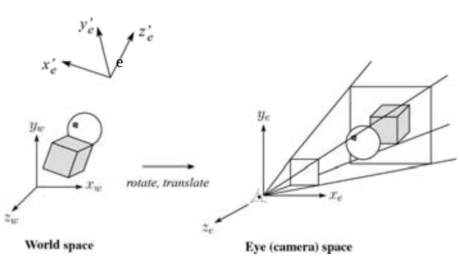


3D Geometry Pipeline (cont’d)



World -> eye transformation

Let’s look at how we would compute the world→eye transformation.



gluLookAt

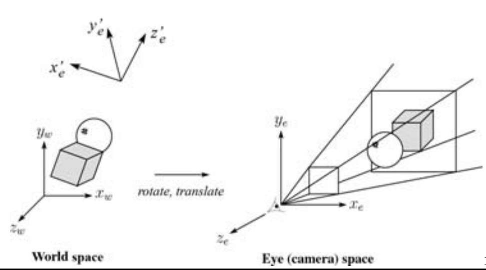
To specify the world->eye transformation, OpenGL has a helper command:

gluLookAt(*eyex*, *eyey*, *eyez*, *atx*, *aty*, *atz*, *upx*, *upy*, *upz*)

To simplify notation, we’ll re-write as:

gluLookAt (**e**, **a**, **u**)

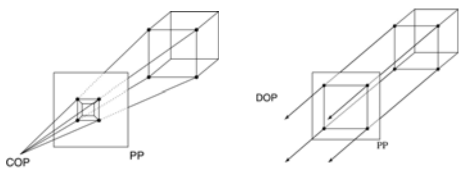
Note that **u** is only required to lie in the $y'_e - z'_e$ plane.



Projections

Projections transform points in *n*-space to *m*-space, where *m*<*n*.

In 3-D, we map points from 3-space to the **projection plane** (PP) (a.k.a., image plane) along **projectors** (a.k.a., viewing rays) emanating from the center of projection (COP):



There are two basic types of projections:

- Perspective – distance from COP to PP finite
- Parallel – distance from COP to PP infinite

Parallel projections

For parallel projections, we specify a **direction of projection** (DOP) instead of a COP.

There are two types of parallel projections:

- **Orthographic projection** – DOP perpendicular to PP
- **Oblique projection** – DOP not perpendicular to PP

We can write orthographic projection onto the *z=0* plane with a simple matrix.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Normally, we do not drop the z value right away. Why not?

Z-buffer

The **Z-buffer** or **depth buffer** algorithm [Catmull, 1974] can be used to determine which surface point is visible at each pixel.

Here is pseudocode for the Z-buffer hidden surface algorithm:

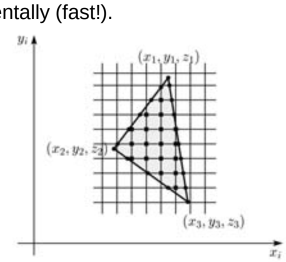
```
for each pixel (i,j) do
    Z-buffer [i,j] ← FAR
    Framebuffer[i,j] ← <background color>
end for
for each polygon A do
    for each pixel in A do
        Compute depth z of A at (i,j)
        if z > Z-buffer [i,j] then
            Z-buffer [i,j] ← z
            Framebuffer[i,j] ← color of A
        end if
    end for
end for
```

Q: What should FAR be set to?

Rasterization

The process of filling in the pixels inside of a polygon is called **rasterization**.

During rasterization, the *z* value can be computed incrementally (fast!).



Curious fact:

- Described as the “brute-force image space algorithm” by [SSS]
- Mentioned only in Appendix B of [SSS] as a point of comparison for huge memories, but written off as totally impractical.

Today, Z-buffers are commonly implemented in hardware.

Properties of parallel projection

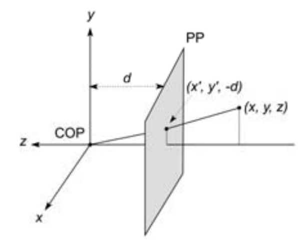
Properties of parallel projection:

- Not realistic looking
- Good for exact measurements
- Are actually a kind of affine transformation
  - Parallel lines remain parallel
  - Ratios are preserved
  - Angles not (in general) preserved
- Most often used in CAD, architectural drawings, etc., where taking exact measurement is important

21

Derivation of perspective projection

Consider the projection of a point onto the projection plane:



By similar triangles, we can compute how much the *x* and *y* coordinates are scaled:

[Note: Angel takes *d* to be a negative number, and thus avoids using a minus sign.]

22

Homogeneous coordinates revisited

Remember how we said that affine transformations work with the last coordinate always set to one.

What happens if the coordinate is not one?

We divide all the coordinates by *w*:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

If $w = 1$, then nothing changes.

Sometimes we call this division step the "perspective divide."

23

Homogeneous coordinates and perspective projection

Now we can re-write the perspective projection as a matrix equation:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix}$$

After division by *w*, we get:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{d}{z}x \\ -\frac{d}{z}y \\ 1 \end{bmatrix}$$

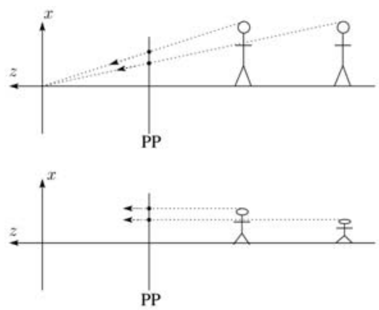
Again, projection implies dropping the *z* coordinate to give a 2D image, but we usually keep it around a little while longer.

24

Projective normalization

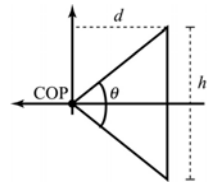
After applying the perspective transformation and dividing by w, we are free to do a simple parallel projection to get the 2D image.

What does this imply about the shape of things after the perspective transformation + divide?



Viewing angle

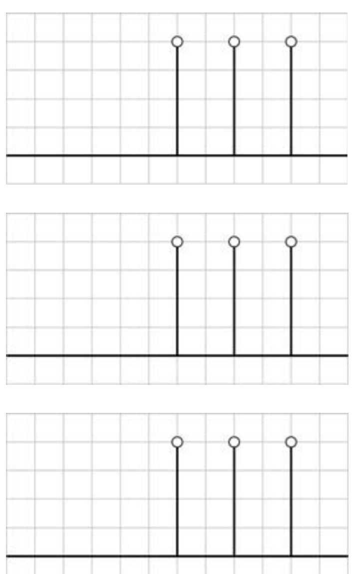
An alternative to specifying the distance from COP to PP is to specify a viewing angle:



Given the height of the image h and θ, what is d?

What happens to d as θ increases (while h is constant)?

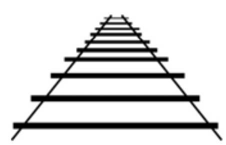
Zoom and dolly



Vanishing points

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

Think of train tracks receding into the horizon...



The 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 transformation we get:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ -(p_z + tv_z)/d \end{bmatrix}$$

Vanishing points (cont'd)

Dividing by w:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} -\dfrac{p_x + tv_x}{p_z + tv_z} d \\ -\dfrac{p_y + tv_y}{p_z + tv_z} d \\ 1 \end{bmatrix}$$

Letting t go to infinity:

We get a point!

What happens to the line **l** = **q** + *t***v**?

Each set of parallel lines intersect at a **vanishing point** on the PP.

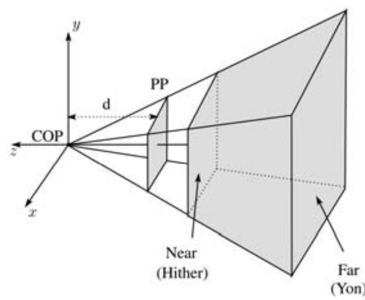
Q: How many vanishing points are there?

29

Clipping and the viewing frustum

The center of projection and the portion of the projection plane that map to the final image form an infinite pyramid. The sides of the pyramid are **clipping planes**.

Frequently, additional clipping planes are inserted to restrict the range of depths. These clipping planes are called the **near** and **far** or the **hither** and **yon** clipping planes.



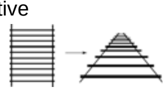
All of the clipping planes bound the the **viewing frustum**.

30

Properties of perspective projections

The perspective projection is an example of a **projective transformation**.

Here are some properties of projective transformations:



- Lines map to lines
- Parallel lines do <u>not</u> necessarily remain parallel
- Ratios are <u>not</u> preserved

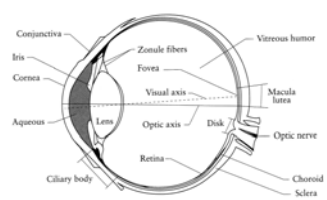
One of the advantages of perspective projection is that size varies inversely with distance – looks realistic.

A disadvantage is that we can't judge distances as exactly as we can with parallel projections.

31

Human vision and perspective

The human visual system uses a lens to collect light more efficiently, but records perspectively projected images much like a pinhole camera.



Q: Why did nature give us eyes that perform perspective projections?

Q: Do our eyes "see in 3D"?

32

Summary

What to take away from this lecture:

- All the boldfaced words.
- An appreciation for the various coordinate systems used in computer graphics.
- How to compute the world->eye coordinate transformation with gluLookAt.
- How a pinhole camera works.
- How orthographic projection works.
- How the perspective transformation works.
- How we use homogeneous coordinates to represent perspective projections.
- The properties of vanishing points.
- The mathematical properties of projective transformations.