Projections

Zoran Popović
CSE 457
Spring 2019

Reading

Optional reading:

- Angel and Shreiner, 5.1-5.6
- Marschner and Shirley, Ch. 7

Further reading:

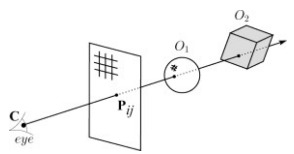
- Foley, et al, Chapter 5.6 and Chapter 6
- David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, 2nd Ed., McGraw-Hill, New York, 1990, Chapter 2.
- I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, A characterization of ten hidden surface algorithms, *ACM Computing Surveys*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

Ray casting

We will be using a pinhole camera model, where all viewing rays pass through a single center of projection.

Our problem now is to figure out where points land on the image plane, and which surface points are visible.

One natural way to do this is with **ray casting**.



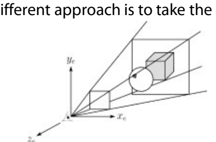
Approach: for each pixel center P_{ij}

- Send ray from eye point (COP), $\mathbf{C}$, through P_{ij} into scene:

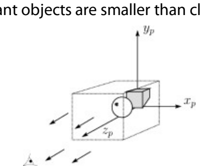
- Intersect ray with each object.
- Given set of intersections $\{t_k\}$, keep the closest one:

Warping space

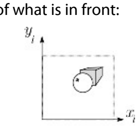
A very different approach is to take the imaging setup:



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



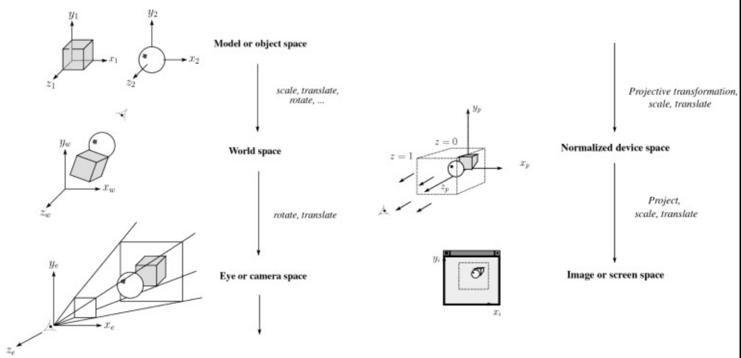
and then just draw everything onto the image plane, keeping track of what is in front:



3D Geometry Pipeline

Graphics hardware follows the "warping space" approach.

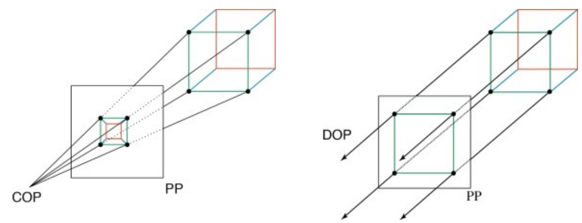
Before being turned into pixels, a piece of geometry goes through a number of transformations...



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 [Straßer, 1974][Catmull, 1974] can be used to determine which surface point is visible at each pixel.

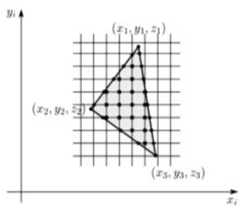
Here is pseudocode for the Z-buffer hidden surface algorithm, for a viewer looking down the $-z$ axis (bigger – i.e., more positive – z's are closer):

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

Q: What should *FAR* be set to?

Z-buffers and 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.

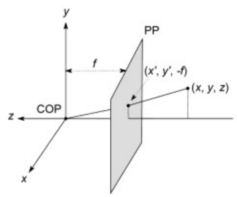
Properties of parallel projection

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

Derivation of perspective projection

Consider the projection of a point onto the projection plane:



The distance f is called the **focal length** of the pinhole camera , as it is essentially equivalent to the focal length of a lens system in a camera.

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

[Note: Angel uses d instead of f and takes it to be a negative number, and thus avoids using a minus sign.]

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."

Homogeneous coordinates and perspective projection

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

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

After division by w, we get:

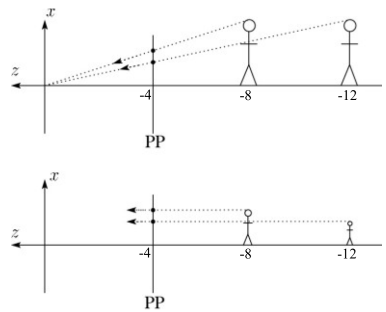
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{f}{z}x \\ -\frac{f}{z}y \\ 1 \end{bmatrix}$$

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

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.

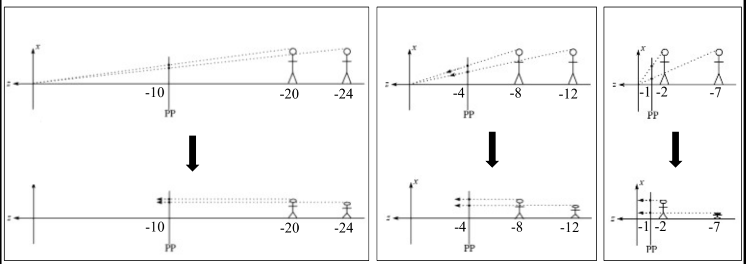
What does this imply about the shape of things after the perspective transformation + divide?



Dolly and zoom

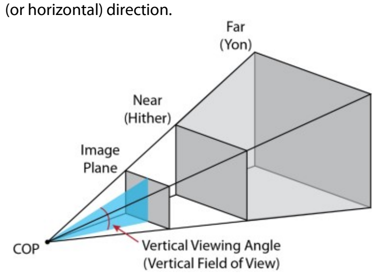
Now imagine starting far away from your subject, with a long focal length ("zoomed in") and then walk toward your subject ("dolly in") while reducing the zoom.

You can keep the subject of constant size on the image plane, but everything in front and behind will change size, sometimes dramatically!



Field of view, clipping, viewing frustum

We can alternatively parameterize focal length in terms of **viewing angle** or **field of view** with respect to the vertical (or horizontal) direction.



The center of projection and the (finite) image form an infinite pyramid. Objects outside the pyramid are not drawn. The sides of the pyramid are **clipping planes**.

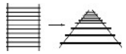
We can also restrict the range of *depths* with **near (hither)** and **far (yon)** clipping planes.

Together, the clipping planes bound the **viewing frustum**.

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 not necessarily remain parallel
- Ratios are not 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.

Summary

What to take away from this lecture:

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