Perspective Projection[1]

Dr. Mihail

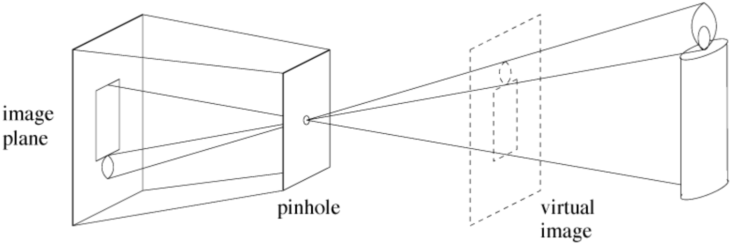
September 6, 2016

[1] Some of the images in these slides are taken from Dr. Stephen Chenney graphics course at UW Madison `http://research.cs.wisc.edu/graphics/Courses/559-s2002/`

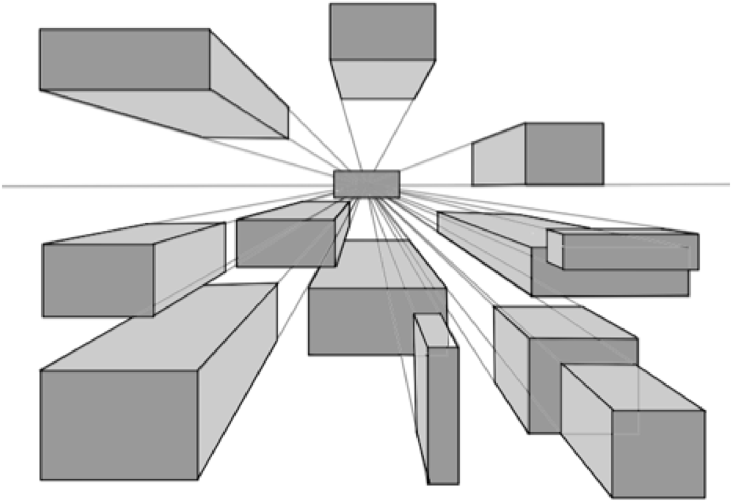
Definitions

- **View Space**: coordinate system with the viewer looking down the -z axis, with +x to the right and +y up
- **World-View Transformation**: takes points in world space and converts them into points in view space
- **Projection Transformation**: takes points in view space and converts them into points in **Canonical View Space**
- **Canonical View Space**: coordinate system with the viewer looking along -z, +x to the right and +y up. Here everything inside the cube x:[-1, 1], y:[-1, 1], z:[-1, 1] using orthogonal projection.

Perspective Projection

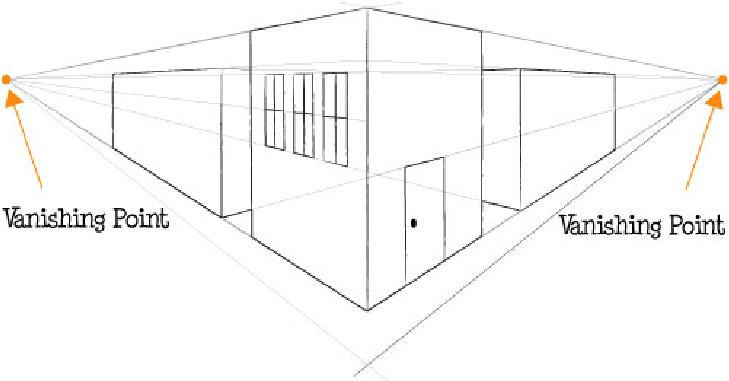


One Point Perspective



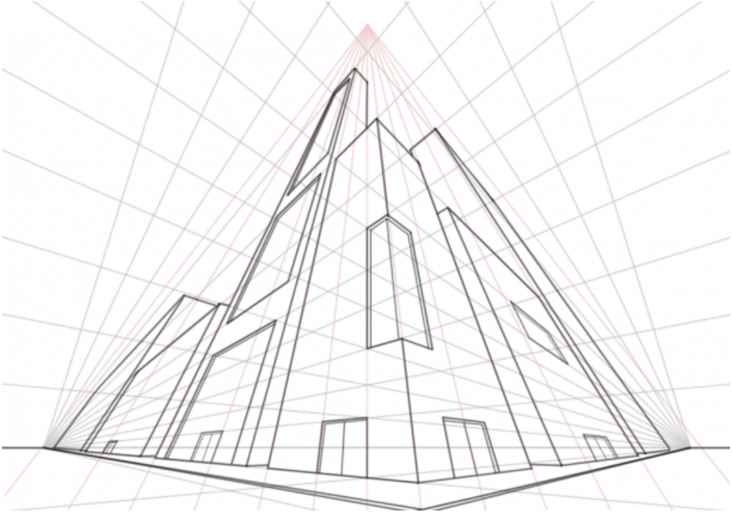
https://www.youtube.com/watch?v=qmSg_F4P5yU

Two Point Perspective



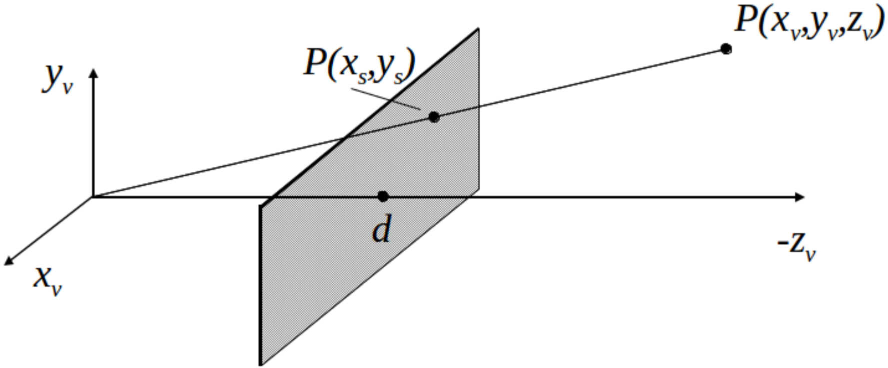
https://www.youtube.com/watch?t=52&v=7ZYBWA-ifEs

Three Point Perspective



https://www.youtube.com/watch?v=BfHRReALvVc

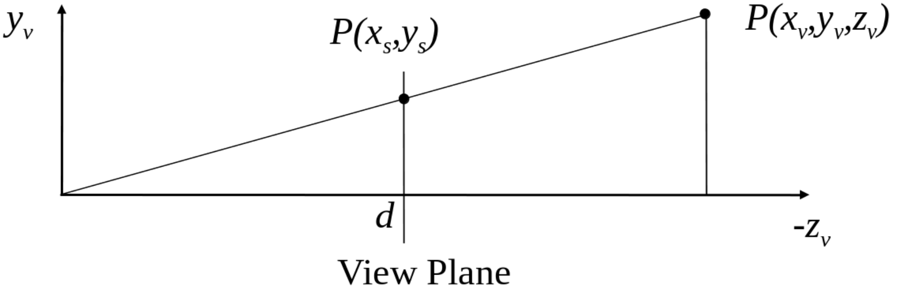
Simple Perspective Transformation



Simple Perspective Transformation

By similar triangles

$$\frac{x_s}{d} = \frac{x_v}{z_v} \qquad \frac{y_s}{d} = \frac{y_v}{z_v}$$



Simple Perspective Transformation

Using homogeneous coordinates

$$\begin{bmatrix} x_s \\ y_s \\ d \end{bmatrix} \equiv \begin{bmatrix} x_v \\ y_v \\ z_v \\ z_v/d \end{bmatrix} \qquad \mathbf{P}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \mathbf{P}_v$$

- One can write a line in parametric form: $x = x_0 + td$
- x_0 is a point on a line, t is a scalar (distance along the line from x_0) and d is a direction (unit length)
- Different x_0 gives different parallel lines

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} + t \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ z_d \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{x_0 + tx_d}{z_0 + tz_d} \\ \frac{y_0 + ty_d}{z_0 + tz_d} \\ 1 \end{bmatrix}$$

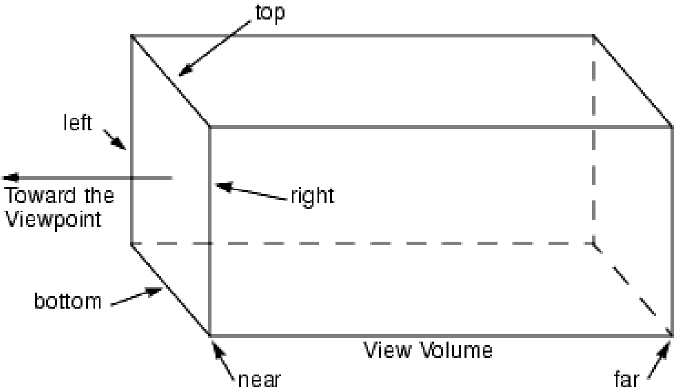
Taking the limit as $t \to \infty$, we get $\begin{bmatrix} \frac{fx_d}{z_d} \\ \frac{fy_d}{z_d} \\ f \end{bmatrix}$

Simple Perspective Transformation

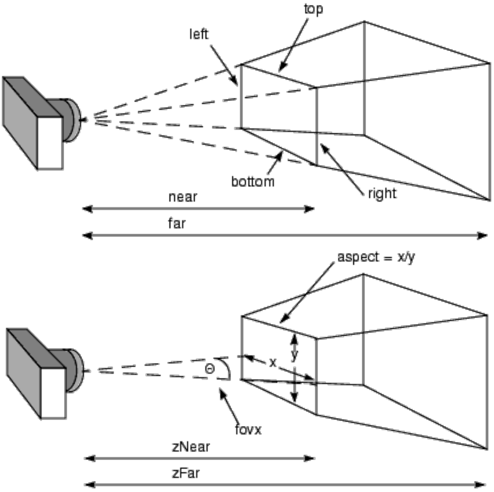
Problems

- This does not map points to a Canonical View Volume
- Insufficient for all applications (e.g., depth testing, because we throw away information)

Orthographic View Volume



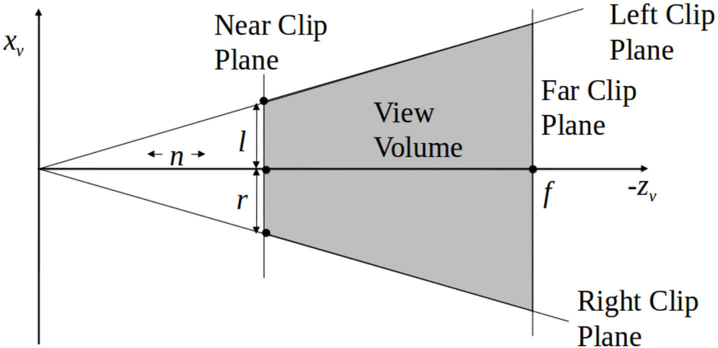
Perspective View Volume



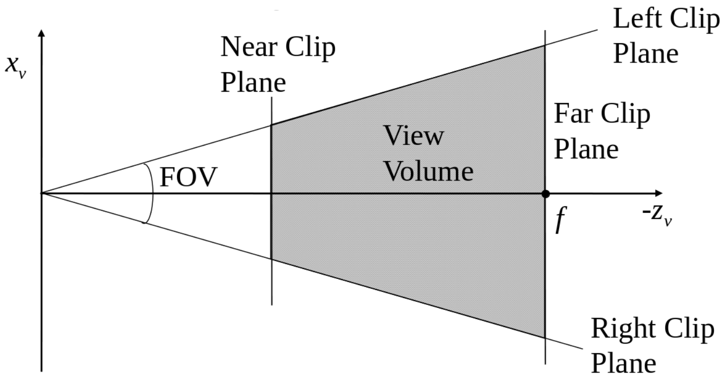
Perspective View Volume

- Near and far planes are parallel to the image plane $z_v = n$, $z_v = f$
- Other planes all pass through the center of projection
- Left and right planes intersect the image planes in vertical lines
- The top and bottom planes intersect the image plane in horizontal lines

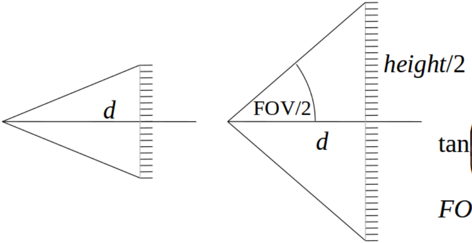
Perspective View Volume



Perspective View Volume



Can convert from image height to FOV or viceversa.

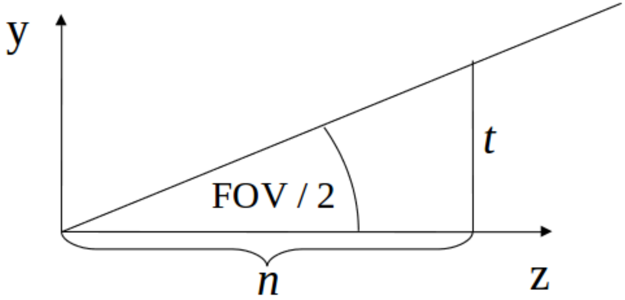


$$\tan\left(\frac{FOV}{2}\right)=\frac{height}{2d}$$

$$FOV=2\tan^{-1}\left(\frac{height}{2d}\right)$$

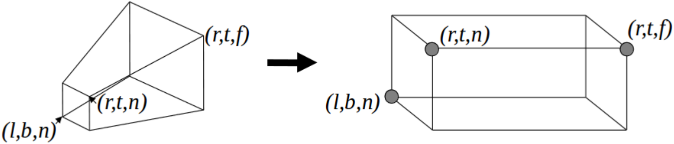
Perspective View Volume

Symmetry.



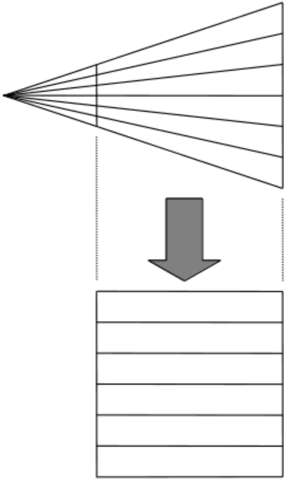
Transformation

We need a matrix that transforms

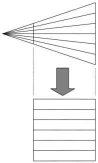


Transformation

We need a matrix that transforms



Transformation



- Convert the perspective case to orthographic so we can use the canonical view space in the existent pipeline
- The intersection of lines with the near clip plane should not change

General Perspective

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{bmatrix} \equiv \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- This matrix leaves points with $z = n$ unchanged
- It maps depth properly
- We can multiply a homogeneous matrix by any number without changing the final point, so the two matrices have the same effect

MV.js perspective()

```
function perspective( fovy, aspect, near, far )
{
    var f = 1.0 / Math.tan( radians(fovy) / 2 );
    var d = far - near;

    var result = mat4();
    result[0][0] = f / aspect;
    result[1][1] = f;
    result[2][2] = -(near + far) / d;
    result[2][3] = -2 * near * far / d;
    result[3][2] = -1;
    result[3][3] = 0.0;

    return result;
}
```

MV.js perspective()

$$f = \frac{1}{tan^{-1}(\frac{fovy}{2})}$$
$$d = far - near$$

The Matrix

$$M_p = \begin{bmatrix} \frac{f}{a} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{-n+f}{d} & \frac{2nf}{d} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$