CS 428: Fall 2009

Introduction to Computer Graphics

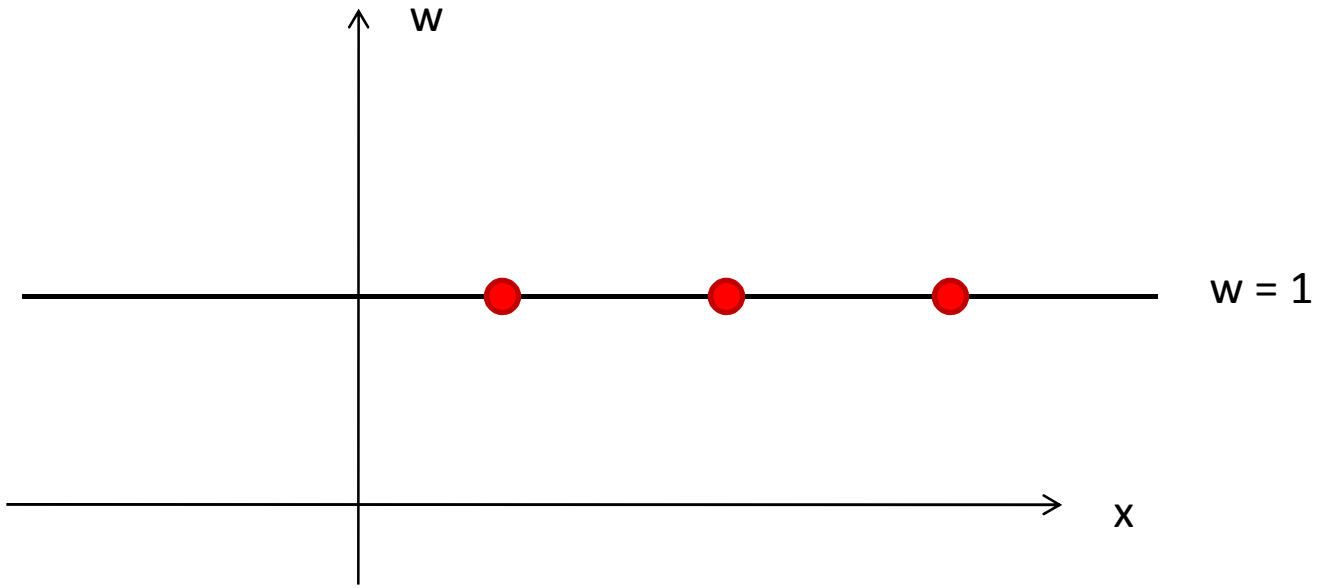
Viewport + clipping

Perspective transformation

Geometric intuition

- Shear other axes along w-axis
- For single vanishing point in 2D: shearing the x-axis a w.r.t. w-axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix}$$

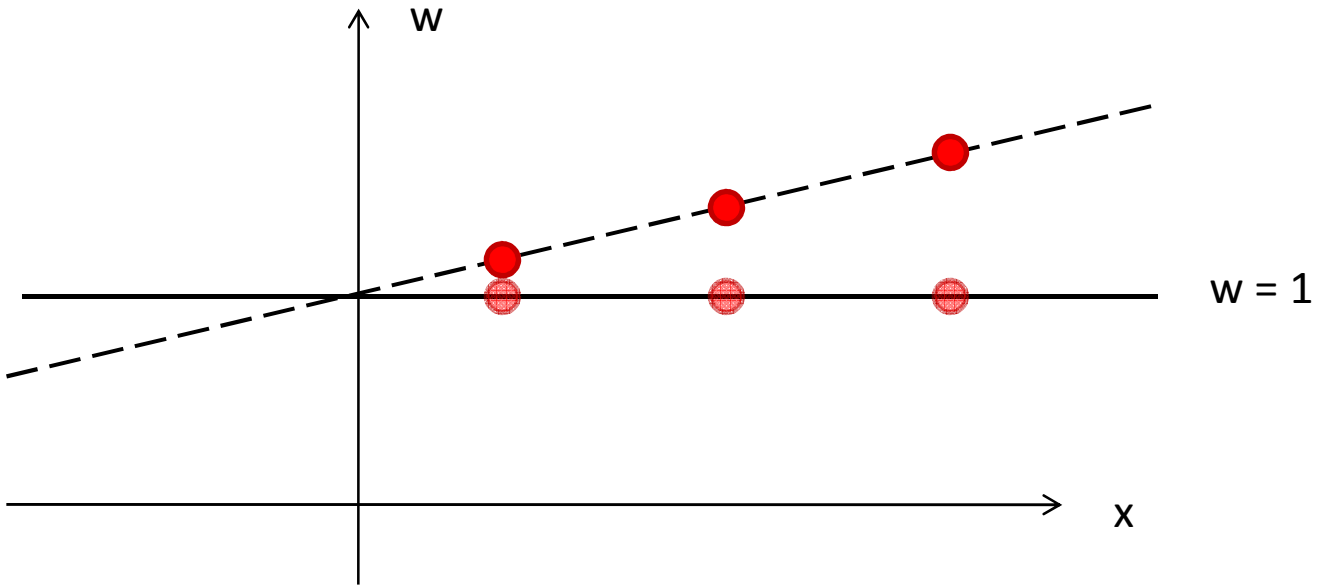


Perspective transformation

Geometric intuition

- Shear other axes along w-axis
- For single vanishing point in 2D: shearing the x-axis a w.r.t. w-axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix}$$

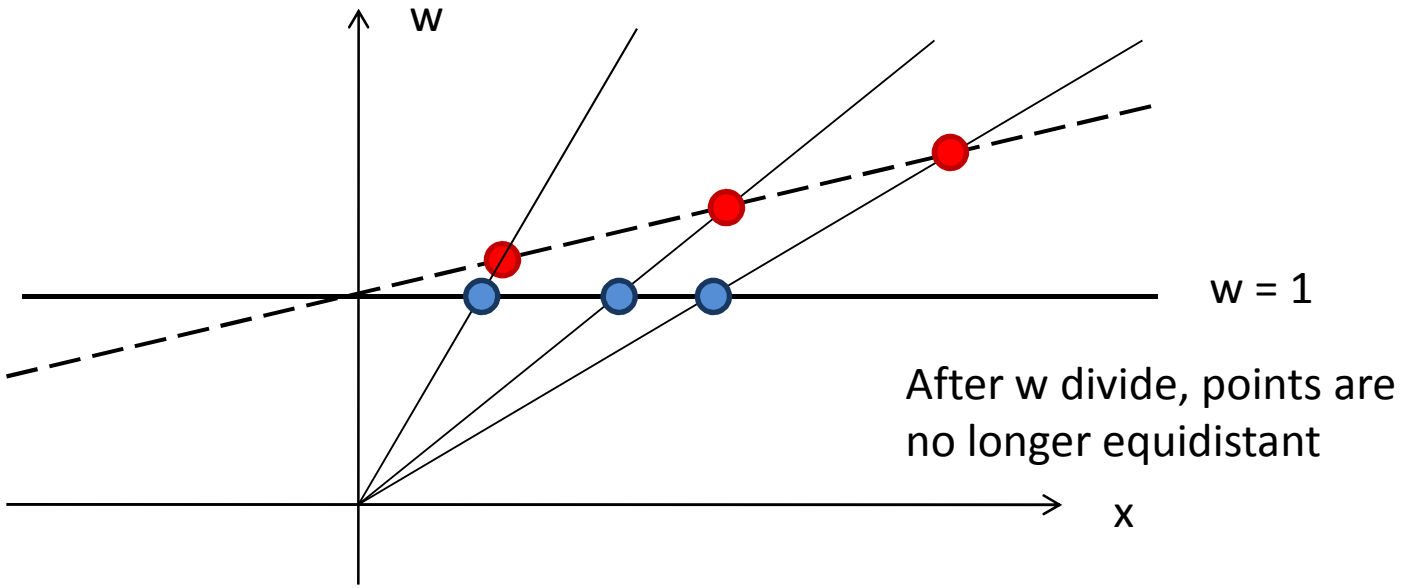


Perspective transformation

Geometric intuition

- Shear other axes along w-axis
- For single vanishing point in 2D: shearing the x-axis a w.r.t. w-axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix}$$



Viewport transformation

- **Normalized device coordinates** (floating point) to **viewport coordinates** (integers, pixels)
- Map one rectangle to the other
- Conceptually includes *left*, *right*, *top*, *bottom* from glFrustum/glOrtho transformation
  - Although this does happen before viewport transformation

In OpenGL code

using **`gluProjection`**

```
projection(int x, int y, int w, int h) {
  float aspect = (float) w / (float) h;
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluProjection(40.0, aspect, 1.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
}
```

- y-axis opening angle (40 degrees) **constant**
- Projection matrix is aspect ratio sensitive

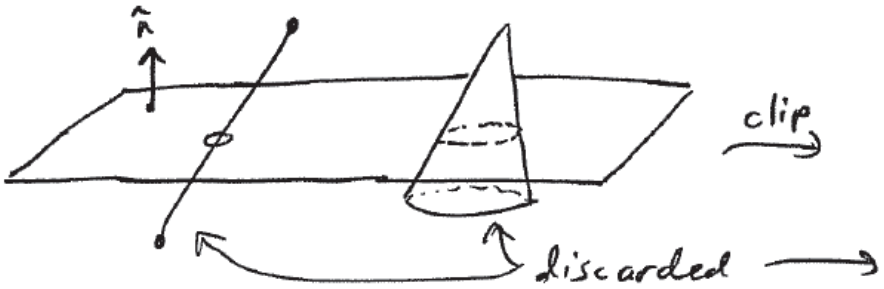
In OpenGL code

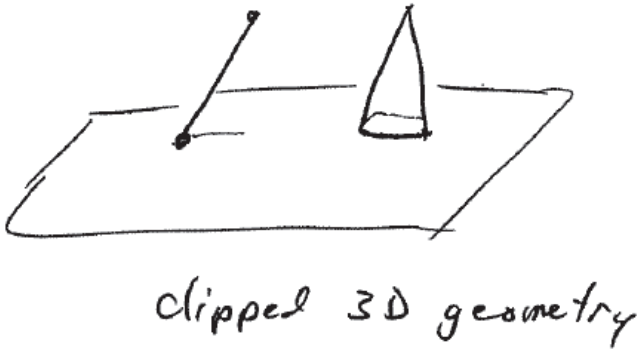
using **`glFrustum`**

```
projection(int x, int y, int w, int h) {
  float aspect = (float) w / (float) h;
  float s2w = 1.0f/800.0f;
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(-w/2*s2w, w/2*s2w, -h/2*s2w,
  h/2*s2w, 1.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
}
```

- Window size **defines** viewport in this case

Clipping

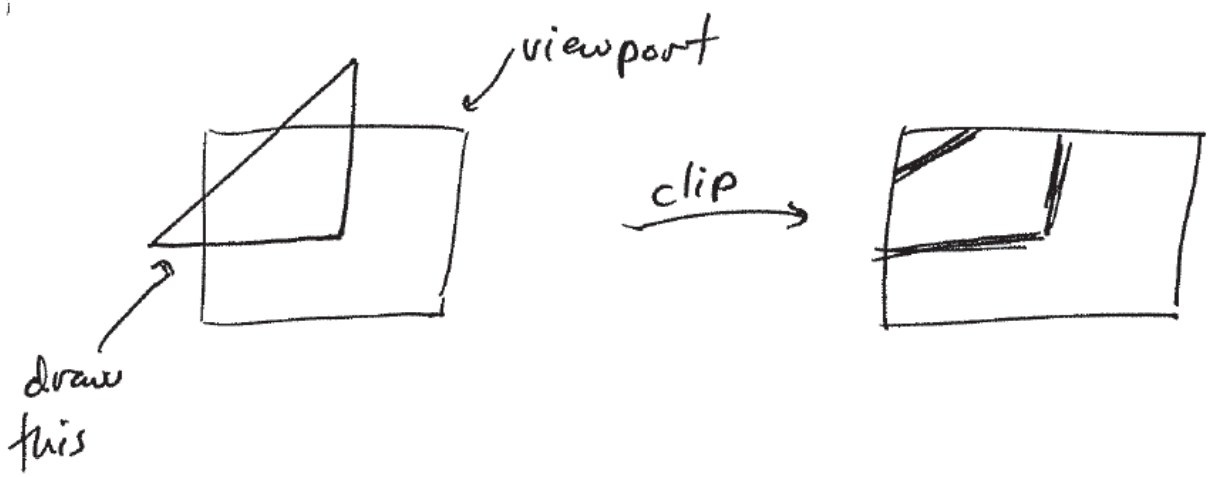




- All 6 faces of the view volume are clip planes
- Why?
  - Sides: not visible geometry
  - Front: too close or behind camera
  - Back: distant objects are small dots

Depth buffer precision!

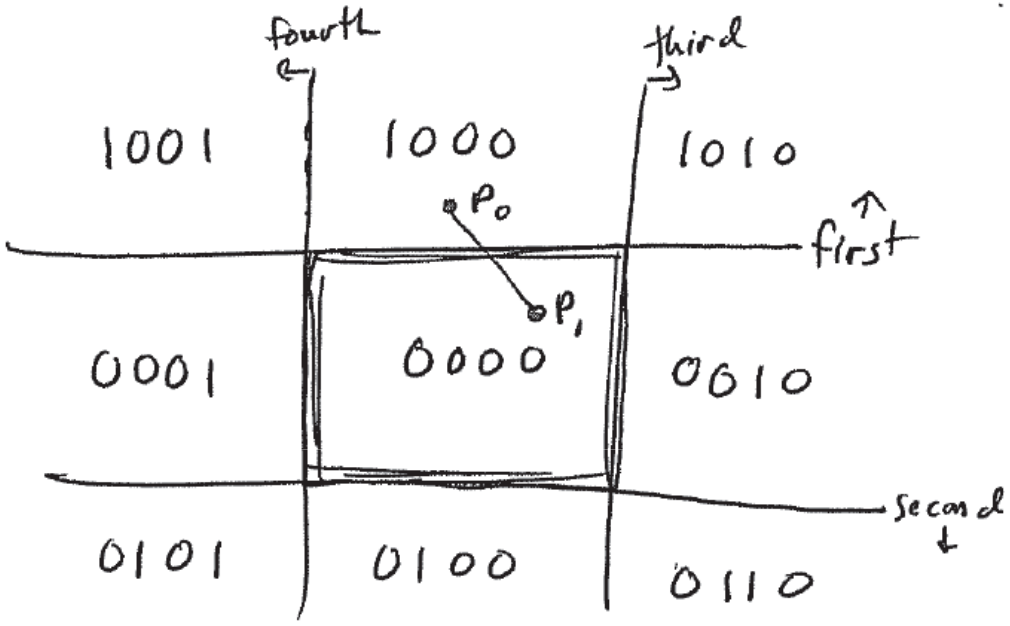
Clipping

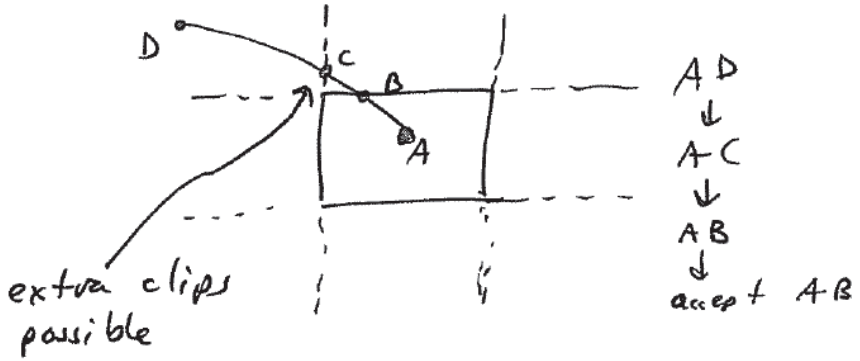


- To a rectangle
- Why?
  - More efficient
  - Outside of frame buffer memory

Cohen-Sutherland line clipping

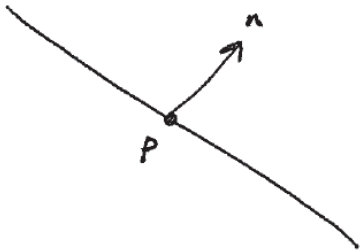
- 4-bit "outcode"
- $OC_1 = OC_2 = 0$
  Trivial accept
- $OC_1 \& OC_2 \neq 0$
  Trivial reject
  (entirely in half-plane)
- Else: divide and conquer
  - Cut line along plane
  - Iterate /w segments





Extra clipping planes

- 6 more planes in OpenGL
- Point normal form [A, B, C, D]
- Transformed like any other geometry



equation of plane:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - p\right) \cdot n = 0$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot n - p \cdot n = 0$$

$n = \begin{bmatrix} A \\ B \\ C \end{bmatrix}$ and $D = -p \cdot n$, then

same as $Ax + By + Cz + D = 0$

```
glClipPlane(GL_CLIP_PLANE0, eqn)
glEnable(GL_CLIP_PLANE0)
...
glDisable(GL_CLIP_PLANE0)
```