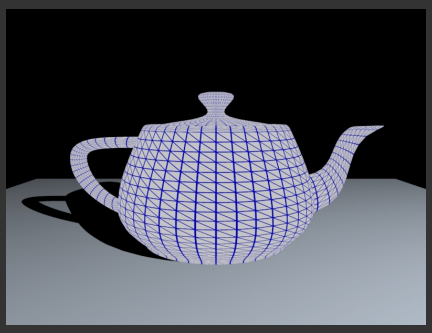
CSE168: Rendering Algorithms
Triangle Rasterization



Henrik Wann Jensen
henrik@cs.ucsd.edu

Overview

- Basic transformations
- Line drawing
- Triangle rasterization
- Perspective correct texturing
- Hidden surface elimination

Basic vector math

A vector:

$$\vec{a} = [x\ y\ z]$$

Dot product:

$$\vec{a} \cdot \vec{b} = ||\vec{a}||\,||\vec{b}||\cos\theta$$

Cross product:

$$\vec{n} = \vec{a} \times \vec{b}$$
$$||\vec{n}|| = ||\vec{a}||\,||\vec{b}||\sin\theta$$

Transformations

A transformation matrix:

$$T = \begin{bmatrix} x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{13} & x_{23} & x_{33} & x_{43} \\ x_{14} & x_{24} & x_{34} & x_{44} \end{bmatrix}$$

A 4D vector:

$$\vec{a} = [x\ y\ z\ w]$$

A point has $w \neq 0$ and a vector has $w = 0$.

Transformations

Transformation of a vector or a point:

$$\vec{b} = T \cdot \vec{a}$$

Transformations

Translation:

$$T_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations

Scaling:

$$T_S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations

Rotation:

$$T_{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations

$$T = \begin{bmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

uvw form a basis for the transformed coordinate system.

Windowing transform

Map $[-1,1] \times [-1,1] \times [-1,1]$ to a screen with $n_x \times n_y$ pixels.

$$T_s = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Simple orthographic projection

Project box with corners $\vec{b}$ and $\vec{t}$

$$T_o = \begin{bmatrix} \frac{2}{\vec{t}_x - \vec{b}_x} & 0 & 0 & 0 \\ 0 & \frac{2}{\vec{t}_y - \vec{b}_y} & 0 & 0 \\ 0 & 0 & \frac{2}{\vec{t}_z - \vec{b}_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{\vec{t}_x + \vec{b}_x}{2} \\ 0 & 1 & 0 & -\frac{\vec{t}_y + \vec{b}_y}{2} \\ 0 & 0 & 1 & -\frac{\vec{t}_z + \vec{b}_z}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic view projection

Camera position: $\vec{e}$
Camera direction: $\vec{d}$
Camera up: $\vec{u_p}$

Orthonormal basis:

$$\vec{w} = -\frac{\vec{d}}{||\vec{d}||} \quad , \quad \vec{u} = \frac{\vec{u_p} \times \vec{w}}{||\vec{u_p} \times \vec{w}||} \quad , \quad \vec{v} = \vec{w} \times \vec{u}$$

Orthographic view projection

$$T_v = \begin{bmatrix} \vec{u}_x & \vec{v}_x & \vec{w}_x & 0 \\ \vec{u}_y & \vec{v}_y & \vec{w}_y & 0 \\ \vec{u}_z & \vec{v}_z & \vec{w}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective transform

$$T_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\vec{b}_z + \vec{t}_z}{\vec{b}_z} & -\vec{t}_z \\ 0 & 0 & \frac{1}{\vec{b}_z} & 0 \end{bmatrix}$$

Multiply by b_z/z through homogenization.

$$T_{per} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z\frac{\vec{b}_z + \vec{t}_z}{\vec{b}_z} - \vec{t}_z \\ \frac{z}{\vec{b}_z} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{\vec{b}_z x}{z} \\ \frac{\vec{b}_z y}{z} \\ \vec{b}_z + \vec{t}_z - \frac{\vec{t}_z \vec{b}_z}{z} \\ 1 \end{bmatrix}$$

Complete transform

$$T = T_s T_o T_p T_v T_m$$

T_m is the model transformation

To transform a vertex $\vec{v}$

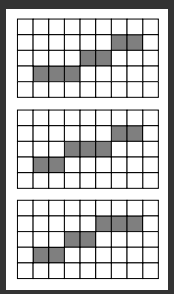
$$\vec{v}_h = T\vec{v}$$

Final pixel position: $(\frac{\vec{v}_{h,x}}{\vec{v}_{h,w}}, \frac{\vec{v}_{h,y}}{\vec{v}_{h,w}})$

A Visual Break

Line drawing

Given 2 points (x_0, y_0) and (x_1, y_1) draw a line between them such that there are no holes in the line.



A 2D line

$$y = ax + b$$

a is the slope of the line.
What if $x = const$?

$$ay + bx + c = 0$$

How to compute the line

Given (x_0, y_0) and (x_1, y_1):

$$a * y_0 + b * x_0 + c = 0$$
$$a * y_1 + b * x_1 + c = 0$$

$$\vec{n} \cdot [x\ y] + c = 0$$
$$\vec{n} = [(y_0 - y_1)\ (x_1 - x_0)]$$

How to compute the line

$$(y_0 - y_1)x + (x_1 - x_0)y + c = 0$$

$$(y_0 - y_1)x + (x_1 - x_0)y + x_0y_1 - y_0x_1 = 0$$

Bresenham's algorithm

J. Bresenham
"Algorithm for Computer Control of a Digital Plotter"
IBM System Journal, pages 25--30, 1965

Drawing the line

Split domain into four quadrants.

The following assumes that:

$$x_1 > x_0 \quad \text{and} \quad (x_1 - x_0) > (y_1 - y_0)$$

Drawing the line

```
y=y0
for x=x0 to x1
  putpixel(x,y)
  if ( d < 0 )
    y = y + 1
```


Drawing the line

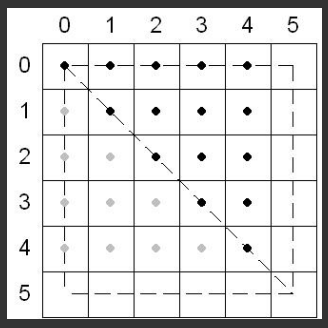
d is the location of the next pixel compared to the line.

$$\begin{aligned} d &= f(x+1, y+0.5) \\ &= (y_0 - y_1)(x+1) + (x_1 - x_0) * (y+0.5) + x_0y_1 - y_0x_1 \\ &= 2(y_0 - y_1)(x+1) + (x_1 - x_0) * (2y+1) + 2x_0y_1 - 2y_0x_1 \end{aligned}$$

Only integers !

Triangle rasterization

Given a triangle with corners (x_0, y_0), (x_1, y_1), and (x_2, y_2) draw a filled triangle (rasterize it).



Triangle rasterization

- Scanconversion
- Active edge lists
- Incremental updates

Shirley's method

```
compute minx, miny, maxx, maxy
for x = minx to maxx
  for y = miny to maxy
    if (x,y) is in triangle
      putpixel( x, y )
```


Scan conversion (top triangles)

```
xleft = xright = x0;
for y=y0 to y1
  drawline( xleft, xright, y )
  xleft += dx_left
  xright += dx_right
```


Scan conversion (top triangles)

$$dx_{left} = \frac{x_1 - x_0}{y_1 - y_0}$$
$$dx_{right} = \frac{x_2 - x_0}{y_2 - y_0}$$

Scan conversion general triangles

Break triangle into a top and a bottom triangle

Interpolating depth, colors etc.

```
t1 = 0
dt1 = 1.0/(y1-y0)
t2 = 0
dt2 = 1.0/(y2-y0)
for y=y0 to y1
  interpolate( value_left, y0, y1, t1 )
  interpolate( value_right, y0, y2, t2 )
  t1 += dt1
  t2 += dt2
  drawline( value_left, value_right,
            x_left, x_right, y )
```


Perspective corrected interpolation

Triangle edge:

$$ax_{per} + by_{per} + c = 0$$

Perspective projection:

$$x_{per} \approx k\frac{x}{z}$$

$$y_{per} \approx k\frac{y}{z}$$

Triangle edge corresponds to:

$$a\,k\frac{x}{z} + b\,k\frac{y}{z} + c = 0$$

Perspective corrected interpolation

```
uz = u0/z
duz = (u1/z - u0/z) / (y1-y0)
zz = 1/z0
dzz = (1/z1 - 1/z0) / (y1-y0)
for y=y0 to y1
  u = uz / zz
  uz += duz
  zz += dzz
  ...
```

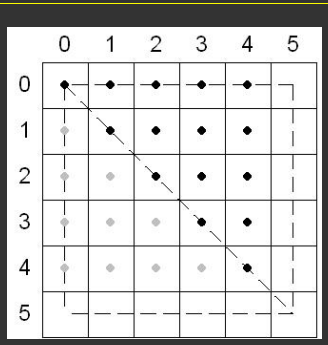

Neighboring triangles

- Use fixed convention (e.g. top-left)

```
y_top = ceil(y0)
y_bottom = ceil(y1) - 1


drawline( xleft, xright, y ) {
  for x = ceil(xleft) to ceil(xright)-1
    putpixel(x,y)
}
```


Neighboring triangles (top-left)



Polygon rasterization

- Keep track of multiple edges
- Break into triangles

Rasterization of 3D triangles

- Convert vertices to 2D screen coordinates
- Eliminate hidden surfaces
- Clipping
- Rasterize

Hidden surface elimination

- Robert's algorithm
- Warnock's algorithm
- Weiler-Atherton
- Catmull's algorithm
- Z-buffer

The Z-buffer

- Keep a float (z-depth) at every pixel

```
plot(x,y,z)
{
  if (z < zbuffer[x,y])
  {
    putpixel(x,y)
    zbuffer[x,y] = z
  }
}
```


Next time

- Ray tracing