

Homework 01

Generating Raster Images

David C. Banks

Electrical Engineering and Computer Science
University of Tennessee

2009

Reading

- ▶ Shirley, Fundamentals of Computer Graphics
- ▶ Chapter 1, pages 1-12
- ▶ Chapter 2, pages 13-51
- ▶ Chapter 3, pages 52-68

Programming

- ▶ Read a geometry file with edges, trapezoids, and triangles
 - ▶ G2
 - ▶ edge 20 10 127 50 10 229
 - ▶ trapezoid 25 15 148 49 50 223 74 15 192 54 50 201
 - ▶ triangle 35 22 148 149 150 102 25 115 48
- ▶ Each triple represents x, y, gray
- ▶ Rasterize the image in pgm format
- ▶ Create webpage with link to tgz
- ▶ Include example geom files and resulting images

Geometry and Shading

- ▶ Geometry is a set of points
- ▶ Shading is the color assigned to the points
- ▶ Computer graphics: apply shading to geometry
- ▶ Powerful technique: linear interpolation

Linear interpolation

- ▶ Estimate $f(t)$, given $f_0=f(t_0)$, $f_1=f(t_1)$
- ▶ Linearly interpolate using points (t_0, f_0) , (t_1, f_1)
 - ▶ `float interpL (float t, float t0, float f0, float t1, float f1)`
 - ▶ `{`
 - ▶ `float df = f1-f0, dt = t1-t0;`
 - ▶ `assert (dt != 0.0);`
 - ▶ `return (f0 + (t-t0) * df / dt);`
 - ▶ `}`

Point structure

- ▶ Basic geometry: a point in \mathbb{R}^2
 - ▶ typedef struct
 - ▶ {
 - ▶ float x, y, gray;
 - ▶ } pointType;
- ▶ Example: pointType p;
 - ▶ p.x = 10; p.y = 20; p.gray = 0;

Edge structure

- ▶ Edge data structure
- ▶ Edge has two points
 - ▶ typedef struct
 - ▶ {
 - ▶ pointType p0, p1;
 - ▶ } edgeType;
- ▶ Example: edgeType e;
 - ▶ e.p0.x = 10; e.p0.y = 20; e.p0.gray = 0;
 - ▶ e.p1.x = 50; e.p1.y = 17; e.p1.gray = 255;

Edge structure

- ▶ Interpolate y as a function of x along the edge
 - ▶ float edgeYofX (float x, edgeType &e)
 - ▶ {
 - ▶ return (interpL (x, e.p0.x, e.p0.y, e.p1.x, e.p1.y));
 - ▶ }
- ▶ Interpolate gray as a function of x along the edge
 - ▶ float edgeGofX (float x, edgeType &e)
 - ▶ {
 - ▶ return (interpL (x, e.p0.x, e.p0.gray, e.p1.x, e.p1.gray));
 - ▶ }

Edge structure

- ▶ Interpolate values along an edge
- ▶ Interpolate x as a function of y along the edge
 - ▶ `float edgeXofY (float y, edgeType &e)`
 - ▶ {
 - ▶ `return (interpL (y, e.p0.y, e.p0.x, e.p1.y, e.p1.x));`
 - ▶ }
- ▶ Interpolate gray as a function of y along the edge
 - ▶ `float edgeGofY (float y, edgeType &e)`
 - ▶ {
 - ▶ `return (interpL (y, e.p0.y, e.p0.gray, e.p1.y, e.p1.gray));`
 - ▶ }

Quadrilateral structure

- ▶ Quadrilateral has two edges
 - ▶ typedef struct
 - ▶ {
 - ▶ edgeType e0, e1;
 - ▶ } quadType;

Trapezoid

- ▶ Trapezoid is a quadrilateral with 2 parallel sides
 - ▶ quadType trap;
 - ▶ trap.e0.p0
 - ▶ trap.e0.p1
 - ▶ trap.e1.p0
 - ▶ trap.e1.p1

Trapezoid

- ▶ On each edge, p_0 is lower than p_1
 - ▶ `assert (trap.e0.p0.y < trap.e0.p1.y);`
 - ▶ `assert (trap.e1.p0.y < trap.e1.p1.y);`
- ▶ Top and bottom edges are horizontal
 - ▶ `assert (trap.e0.p0.y == trap.e1.p0.y);`
 - ▶ `assert (trap.e0.p1.y == trap.e1.p1.y);`
- ▶ Edge e_0 is not right of edge e_1
 - ▶ `assert (trap.e0.p0.x <= trap.e1.p0.x);`
 - ▶ `assert (trap.e0.p1.x <= trap.e1.p1.x);`
 - ▶ Triangle becomes special case of trapezoid

Trapezoid

- ▶ Make spans from left side to right side of trapezoid
 - ▶ edgeType span;
 - ▶ for (yi = ceil(trap.e0.p0.y); yi < floor(trap.e0.p1.y); yi++)
 - ▶ {
 - ▶ span.p0.y = yi;
 - ▶ span.p0.x = edgeXofY (yi, trap.e0)
 - ▶ span.p0.gray = edgeGofY (yi, trap.e0)
 - ▶ span.p1.y = yi;
 - ▶ span.p1.x = edgeXofY (yi, trap.e1)
 - ▶ span.p1.gray = edgeGofY (yi, trap.e1)
 - ▶ edgeH(span); *inner loop; check for coincident points*
 - ▶ }

Trapezoid

- ▶ "Bi-linear" interpolation
 - ▶ linearly interpolate gray value on each edge
 - ▶ linearly interpolate gray value on span
- ▶ Note: "bi-linear" is not linear
- ▶ Linear map $f(u)$ satisfies 2 properties
 - ▶ $f(a*u) = a*f(u)$
 - ▶ $f(u+v) = f(u)+f(v)$

Triangle

- ▶ Triangle has points $p[0]$, $p[1]$, $p[2]$
- ▶ Permute the indexes in x-order, y-order
- ▶ Sort vertexes to produce $p[sx[0]]$, $p[sx[1]]$, $p[sx[2]]$
 - ▶ $p[sx[0]]$ is leftmost, $p[sx[2]]$ is rightmost
- ▶ Sort vertexes to produce $p[sy[0]]$, $p[sy[1]]$, $p[sy[2]]$
 - ▶ $p[sy[0]]$ is bottommost, $p[sy[2]]$ is topmost

Triangle

- ▶ Ex: $p[0] = (200, 100)$, $p[1] = (120, 50)$, $p[2] = (40, 150)$
 - ▶ $p[2]$ has smallest x , so $sx[0] = 2$
 - ▶ $p[0]$ has largest x , so $sx[2] = 0$
 - ▶ $p[1]$ has smallest y , so $sy[0] = 1$
 - ▶ $p[2]$ has largest y , so $sy[2] = 2$

Triangle

- ▶ Split lowest-to-highest edge using middle point
 - ▶ Note: $p[sy[0]]$ is topmost, $p[sy[2]]$ is bottommost
- ▶ $eLowestToHighest.p0 = p[sy[0]];$
- ▶ $eLowestToHighest.p1 = p[sy[2]];$
- ▶ $pSplit.y = p[sy[1]].y;$
- ▶ $pSplit.x = edgeXofY (yi, eLowestToHighest)$
- ▶ $pSplit.g = edgeGofY (yi, eLowestToHighest)$

Triangle

- ▶ Create upper trapezoid
 - ▶ Left : from top $p[sy[2]]$ to $\text{leftmost}(pSplit, p[sy[1]])$
 - ▶ Right: from top $p[sy[2]]$ to $\text{rightmost}(pSplit, p[sy[1]])$
- ▶ Create lower trapezoid
 - ▶ Left : from $\text{leftmost}(pSplit, p[sy[1]])$ to bottom $p[s[0]]$
 - ▶ Right: from $\text{rightmost}(pSplit, p[sy[1]])$ to bottom $p[s[0]]$
- ▶ Draw pixels in each trapezoid

Clipping

- ▶ Bounds-checking for offscreen triangle
 - ▶ If a pixel is offscreen, don't draw it
 - ▶ If a span is offscreen, don't draw it
 - ▶ If a trapezoid is offscreen, don't draw it
 - ▶ If a triangle is offscreen, don't draw it
- ▶ Clipping a triangle
 - ▶ Find the intersection $T \cap R$ of triangle T , rectangle R
 - ▶ Result may not be a triangle
 - ▶ Visit each rectangle edge, slice part of triangle
 - ▶ Re-triangulate into subtriangles
 - ▶ Render

Linear Equations

- ▶ The equation $y = mx+b$ can be written $Ax+By+C=0$
 - ▶ Given two points (x_0,y_0) , (x_1,y_1) , find coefficients A , B , C
 - ▶ $[A \ B] \begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix} = C[1 \ 1]$
 - ▶ $[A \ B] = C[1 \ 1] \begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}^{-1}$
 - ▶ Recall that determinant $\det\left(\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}\right) = x_0 y_1 - x_1 y_0$
 - ▶ Inverse $\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}^{-1} = 1/\det\left(\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}\right) \begin{bmatrix} y_1 & -x_1 \\ -y_0 & x_0 \end{bmatrix}$
 - ▶ Free parameter: choice of C (including sign)
 - ▶ Sign convention: $Ax + By + C = 0$ at **other** vertex

Half spaces

- ▶ Halfplane H: all points $p=(x,y)$ where $Ax+By+C \geq 0$
- ▶ Each triangle edge e_0, e_1, e_2 defines a half plane H_0, H_1, H_2
- ▶ At $p=(x,y)$ inside triangle, $Ax+By+C \geq 0$ for each triple A, B, C
- ▶ Triangle $T = H_0 \cap H_1 \cap H_2$
 - ▶ for ($y_i = y_{Min}; y_i < y_{Max}; y_i++$)
 - ▶ for ($x_i = x_{Min}; x_i < x_{Max}; x_i++$)
 - ▶ if ($pointInTriangle (x_i, y_i, T)$)
 - ▶ draw($x_i, y_i, color(T, x_i, y_i)$)

Linear interpolation

- ▶ Gray value $g = Ax + By + C$
 - ▶ Given three samples $g(x_0, y_0)$, $g(x_1, y_1)$, $g(x_2, y_2)$, find A , B , C
 - ▶ Given three samples $g(x_0, y_0)$, $g(x_1, y_1)$, $g(x_2, y_2)$, find A , B , C
 - ▶ $[A \ B \ C] \begin{bmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix} = [g_0 \ g_1 \ g_2]$
 - ▶ $[A \ B \ C] = \begin{bmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix}^{-1} [g_0 \ g_1 \ g_2]$

Linear interpolation

- ▶ Linear interpolation of gray level
 - ▶ for (yi = yMin; yi < yMax; yi++)
 - ▶ for (xi = xMin; xi < xMax; xi++)
 - ▶ if (pointInTriangle (xi, yi, T))
 - ▶ draw(xi, yi, A*xi + B*yi + C);

k-simplex

- ▶ Simplex
- ▶ 0-simplex is a point $p[0]$
- ▶ 1-simplex is a segment with vertices $p[0], p[1]$
- ▶ 2-simplex is a triangle with vertices $p[0], p[1], p[2]$
- ▶ 3-simplex is a tetrahedron with vertices $p[0] .. p[3]$
- ▶ k-simplex is the convex hull with vertices $p[0] .. p[k]$
 - ▶ Convex hull of $\{p[i]\}$: Intersection of half-spaces of $\{p[i]\}$

k-cell

- ▶ 0-cell c_0 is a point $p[0]$
- ▶ 1-cell c_1 is a segment connecting pair of 0-cells $c_0[0]$, $c_0[1]$
- ▶ 2-cell c_2 is a quadrilateral connecting pair of 1-cells $c_1[0]$, $c_1[1]$
- ▶ 3-cell c_3 is a hexahedron connecting pair of 2-cells $c_2[0]$, $c_2[1]$
- ▶ k-cell c_k is a polytope connecting pair of (k-1)-cells $c_{k-1}[0]$, $c_{k-1}[1]$