Generating Raster Images

Segments and Triangles

David C. Banks

Electrical Engineering and Computer Science
University of Tennessee

2009

Point in line

The equation y = mx+b can be written Ax+By+C=0

Given two points $(x_0, y_0), (x_1, y_1)$ on edge e, find A, B, C

Decide if a point is on line

```
int pointInLine ( float x, float y, edgeType *e )
{
float A, B, C;
coeffsFromPoints ( A, B, C, e->p0, e->p1 );
return ( A*x + B*y + C == 0 )
}
```

Segment as implicit function

Locus of points {(x,y): A*x + b*y + C = 0 }
Restrict to bounding box

```
for ( yi = yMin; yi < yMax; yi++ )
  for ( xi = xMin; xi < xMax; xi++ )
    if ( pointInLine ( xi, yi, segment ) )
      draw( xi, yi, color );
```

Order O(dx*dy) vs O(|(dx,dy)|)
Integer-valued samples may miss the segment
Like playing game of "battleship"

Linear Equations

Affine equation in two variables: $Ax_0 + By_0 + C = 0$

$Ax_0 + By_0 = -C$

$Ax_1 + By_1 = -C$

Without loss of generality, let C=1

$[A\ B] \begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix} = -[1\ 1]$

$[A\ B] \qquad = -[1\ 1] \begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}^{-1}$

Inverse $\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}^{-1} = \begin{bmatrix} y_1 & -x_1 \\ -y_0 & x_0 \end{bmatrix} / \det\left(\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}\right)$

Determinant $\det\left(\begin{bmatrix} x_0 & x_1 \\ y_0 & y_1 \end{bmatrix}\right) = x_0 y_1 - x_1 y_0$

Linear Equations

Example: points (1,3), (2,5)

A*1 + B*3 = -1

A*2 + B*5 = -1

$[A\ B] = -[1\ 1] \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}^{-1}$

Inverse $\begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}^{-1}$ = $\begin{bmatrix} 5 & -2 \\ -3 & 1 \end{bmatrix}$ / det$\left(\begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}\right)$

Determinant det$\left(\begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}\right)$ = 1*5 - 2*3 = -1

$[A\ B] = -1 * [1\ 1] \begin{bmatrix} 5 & -2 \\ -3 & 1 \end{bmatrix} /(-1)$

$[A\ B] = [1\ 1] \begin{bmatrix} 5 & -2 \\ -3 & 1 \end{bmatrix}$ = [5-3 -2+1] = [2 -1]

2x -1y + 1 = 0 , so y = 2x + 1

Linear Equations

Free parameter: A*x/k + B*y/k + C/k = 0

Idea: normalize by magnitude of gradient of f(x,y)

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = \left(\frac{\partial(Ax+By+C)}{\partial x}, \frac{\partial(Ax+By+C)}{\partial y}\right) = (A, B)$$

Let k = $|\nabla f(x,y)| = \sqrt{A^2 + B^2}$

Use A' = A/k, B' = B/k, C' = 1/k

The value | A'x + B'y + C' | is distance from (x, y) to line

Segment as implicit function

Thickened segment through points p_0, p_1: width 2*δ
Locus of points $\{(x, y) : |A * x + B * y + C| < \delta\}$
Restrict to bounding box $\pm\delta$

```
for ( yi = yMin-delta; yi < yMax-delta; yi++ )
  for ( xi = xMin-delta; xi < xMax-delta; xi++ )
    if  ( pointInThickLine ( xi, yi, p0, p1, δ ) )
      draw( xi, yi, color );
```

Random sampling

Points (xi, yi) can be chosen randomly

```
int n, numSamples = ..;
int maxIndex = xRes*yRes;
int xi, yi;
for ( n = 0; n < numSamples; n++)
  {
  coordFromIndex ( &xi, &yi, index );
  index = random() % maxIndex;
    if ( pointInThickLine ( xi, yi, p0, p1, δ ) )
      draw( xi, yi, color );
  }
```

Recursive subdivision of segment

Segment through points p_0, p_1 with recursive subdivision

```
{

point midpoint = 0.5*p0 + 0.5*p1;
draw ( &midpoint );
if ( samePixel( p0, p1 ) return;
subdivide ( p0, &midpoint );
subdivide ( &midpoint, p1 );
}
```

Recursive subdivision of segment

Segment through points p_0, p_1 with recursive subdivision

```
{

point midpoint = 0.5*p0 + 0.5*p1;
draw ( &midpoint );
if ( samePixel( p0, p1 ) return;
subdivide ( p0, &midpoint );
subdivide ( &midpoint, p1 );
}
```

Recursive subdivision of segment

Segment through points p_0, p_1 with random subdivision

```
{
float t = drand48();
point midpoint = (1-t)*p0 + t*p1;
draw ( &midpoint );
if ( samePixel( p0, p1 ) return;
subdivide ( p0, &midpoint );
subdivide ( &midpoint, p1 );
}
```

Segment as parametric equation

Segment through points p_0, p_1

```
int n, numSamples = ..;
float t, dt = length( edge )/(numSamples-1);
point p;
for ( n = 0; n < numSamples; n++ )
  {
  t = n*dt;
  p = (1.0-t)*edge->p0 + t*edge->p1;
  draw ( p );
  }
```

Barycentric coordinates

Bary = weight, heaviness; barycenter = center of mass
Given points p_0, p_1 choose weighted average
$p = a * p_0 + b * p_1, a \geq 0$ and $b \geq 0$
If (a+b == 1), point p $\in$ segment
How to find (a,b) such that a+b=1
Choose b $\in$ [0,1], then a = 1-b

Random sampling

Parameter t can be chosen randomly

```
int n, numSamples = ..;
int xi, yi;
for ( n = 0; n < numSamples; n++)
  {
  t = drand48();
  p = (1.0-t)*edge->p0 + t*edge->p1;
  draw ( p );
  }
```

Halfplane

Halfplane H: all points p=(x,y) where Ax+By+C >= 0

Operator ∂ returns the boundary of a set

Halfplane from edge $\{p_0, p_1\} \in \partial$(H), point $p_2 \in$ H

```
int pointInHalfplane ( pointType *p, edgeType *e, pointType *pIn )
{
for ( xi = xMin; xi < xMax; xi++ )
  if  ( pointInTriangle ( xi, yi, T ) )
    draw( xi, yi, color(T,xi,yi) )
}
```

Region

Constructive Solid Geometry (CSG)

Region $R = R_0 \cap R_1$

Region $R = R_0 \cup R_1$

Region $R = R_0 - R_1 = R_0 \cap \sim R_1$

Example: Intersection of halfplanes $H_0 \cap H_1$

```
if ( pointInHalfplane ( &p, &e0, &pIn0 ) )
  if ( pointInHalfplane ( &p, &e1, &pIn1 ) )
    draw( p );
```

Half planes

Triangle T: intersection of halfplanes
Triangle T = $H_0 \cap H_1 \cap H_2$
3 edges e_0, e_1, e_2 with interior points
Point p $\in$ T satisfies p $\in$ H_0 and p $\in$ H_1 and p $\in$ H_2

```
for ( yi = yMin; yi < yMax; yi++ )
  for ( xi = xMin; xi < xMax; xi++ )
    if  ( pointInTriangle ( xi, yi, T ) )
      draw( xi, yi, color );
```

Linear interpolation

Shading: gray value g = Ax + By + C

Given 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

Cell
0-cell c_0 is a point
1-cell c_1 is a segment connecting pair of 0-cells
2-cell c_2 is a quadrilateral connecting pair of 1-cells
3-cell c_3 is a hexahedron connecting pair of 2-cells
k-cell c_k is a polytope connecting pair of (k-1)-cells

Polytope

Polytope
Polyline is a union of 1-simplexes (segments)
Polygon is a union of 2-simplexes (triangles)
Polyhedron is a union of 3-simplexes (tetrahedra)
Polytope is a union of k-simplexes