Triangle Rasterization

Projection of points
Projection of triangles

Rasterization of triangles

• Determine all pixels covered by the triangle and color them appropriately
  – a pixel is covered by a triangle if its center is inside the triangle
Rasterization of triangles

Rasterization of triangles
Rasterization of triangles
Rasterization of triangles

Finding interior pixels

• Several methods
  – edge equations method
  – DDA (Digital Differential Analyzer) method
Edge equations rasterization

1. bounding box
Edge equations rasterization

1. bounding box

Edge equations rasterization

2. use edges
Edge equations rasterization

2. use edges
Edge equations rasterization

Edge equation

\[ x(y_2 - y_1) - y(x_2 - x_1) - x_1y_2 + y_1x_2 = 0 \]
Edge sidedness

\[ E(x, y) = ax + by + c = x(y_2 - y_1) - y(x_2 - x_1) - x_1y_2 + y_1x_2 \]

\[ E(x_3, y_3) = x_3(y_2 - y_1) - y_3(x_2 - x_1) - x_1y_2 + y_1x_2 \]

\[ E(x_3, y_3) < 0 \]
Triangle rasterization
implementation guidelines

float x[3], y[3];  // image space coordinates of the 3 vertices in pixels (input)
float a[3], b[3], c[3];  // a, b, c for the 3 edge expressions
// establish the three edge equations
// edge that goes through vertices 0 and 1
a[0] = y[1]−y[0]; b[0] = x[1]−x[0]; c[0] = −x[0]*y[1] + y[0]*x[1];
float sidedness;  // temporary variable used to establish correct sidedness
sidedness = a[0]*x[2] + b[0]*y[2] + c[0];
if (sidedness < 0) {
    a[0] = −a[0]; b[0] = −b[0]; c[0] = −c[0];
}
// similar for the other two edges
// compute screen axes-aligned bounding box for triangle
float bbox[2][2];  // for each x and y, store the min and max values
ComputeBBox(x, y, bbox);
ClipBBox(bbox, 0, w, 0, h);
int left = (int) (bbox[0][0] + .5), right = (int) (bbox[0][1] - .5);
int top = (int) (bbox[1][0] + .5), bottom = (int) (bbox[1][1] - .5);

...