Polygon Fills as well as Vertex and Pixel Arrays

CS116A
Chris Pollett
Sep13, 2004.

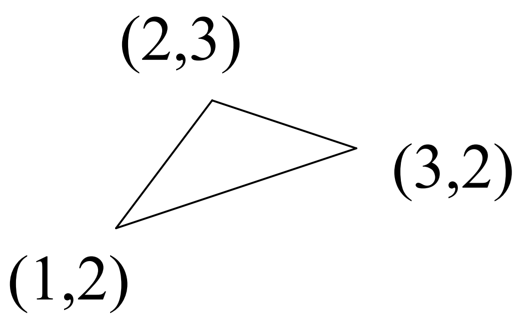
Introduction

Today we’re going to talk about

- Fill Areas
  - Types of Polygons
  - Splitting Concave Polygons
  - Splitting Convex Polygons into Triangles
  - Inside-Outside Tests
  - Polygon Tables
  - Plane Equations
  - Front and Back Faces
  - OpenGL
- Vertex Arrays
- Pixel Arrays

Types of Polygons

- Polygons are sequences of three or more non-collinear vertices in the place. Ex. ((1,2), (2,3), (3,2))

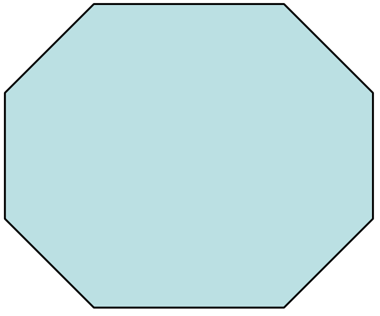


- Notice join last point back to first. Usually require edges to have at most vertices in common

Polygon Classifications

- Look at interior angle formed by adjacent edges. If this angle is always less than 180 then polygon called *convex*, otherwise *concave*.

Concave

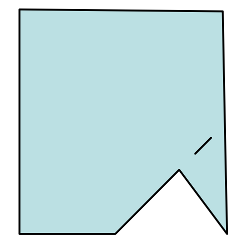


Convex

Identifying Concave Polygons

- Fill algorithms for convex regions easier, so would like an easy algorithm for identifying concave regions.
- If region is convex then the cross-product of adjacent edges will always be of the same sign.
- Make sure to use edges not vertices. E_k given by $V_{k+1} - V_k$.

Splitting Concave Polygons Example



E_1 = (1, 0, 0)   E_2=(1,1,0)

E_3 = (1, -1, 0)   E_4=(0,2,0)

E_5 = (-3, 0, 0)   E_6 =(0, -2, 0)

E_1 x E_2 = (0, 0, 1)   E_2 x E_3 = (0, 0, -2)

E_3 x E_4 = (0, 0, 2)   E_4 x E_5 = (0, 0, 6)

E_5 x E_6 = (0, 0, 6)   E_6 x E_1 = (0, 0, 2)

Since E_2 x E_3 is negative, split the polygon along the line of vector E_2. Use line equation to figure where intersect other polygon edge to split polygon into two pieces

Splitting Convex Polygons into Triangles

- Since triangles are sometimes easier to draw could then split convex polygon into triangles. To do this make any sequence of three consecutive vertices a new triangle. Then delete the middle vertex from the original list of vertices.

Inside-Outside Tests

- To do filling often want to know what is the inside and what is the outside region of a figure.
- **Odd-Even rule:** let (x,y) be the point we are trying to determine if it is inside or outside of an object. Draw a line between this point and a distant point P. If the number of edges of the polyline it crosses is odd then it is an interior point.
- **Nonzero Winding Number Rule:** Draw a line between a (x,y) and P. Now add sum of signs of cross-products of this line with the lines it crosses. If sum is nonzero then is an interior point.

Polygons Tables

- Typically polygons are used in rendering 3D objects. To do this it is convenient to arrange data into three tables: A list of vertices. A list of edges specified as pairs of elements from the first list. A list of polygons specified as sequence of elements from the edge list.

Plane Equations

- In a 3D scene each polygon will live in some plane. So useful to know a little about planes. General equation is:

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

- Can write as: $(A/D)x+(B/D)y+(C/D)z = -1$
- Let $A' =(A/D)$, define B' and C' similarly. Then given three points can solve for these values.
- A normal to the place is the vector (A,B,C)

Front and Back Faces

- The side of a polygon that faces into the interior of a 3D object called a *back face*. Other side called *front face*.
- Given a polygon, let $Ax+By+Cz+D=0$ be its plane. Then a point (x,y,z) is behind the plane if for $Ax+By+Cz+D <0$. If >0 then in front of plane.

OpenGL

- Can draw rectangles with:

```
int vertex1[] = {200, 100};
int vertex2[] = {50, 250};
glRectiv(vertex1, vertex2);
```

- For more general shapes easier to use glBegin, glEnd with one of GL_POLYGON, GL_TRIANGLES, GL_QUADS, GL_TRIANGLES_STRIP, GL_TRIANGLES_FAN, GL_QUAD_STRIP, GL_QUAD_FAN

Vertex Arrays

- Useful to have a way to store list of points that make up an object:

```
typedef GLint vertex3[3];
vertex3 pt[8] = {{0,0,0},{0,1,0},{1,0,0},{1,1,0},
{0,0,1},{0,1,1},{1,0,1},{1,1,1}};
```

- Above could be used for a cube.
- To plot faces can make calls beginning with either glBegin(GL_POLYGON) or glBegin(GL_QUADS)

Vertex Arrays cont'd

- This would require many OpenGL function calls.
- To alleviate this problem use Vertex Arrays:

  glEnableClientState(GL_VERTEX_ARRAY);

  glVertexPointer(3,GL_INT,0,pt);

  GLubyte vertIndex[] = (6,2,3,7, 5,1,0,4, 7,3,1,5, 4,0,2,6, 2,0,1,3, 7,5,4,6);

  glDrawElements(GL_QUADS, 24, GL_UNSIGNED, vertIndex);

- Vertex arrays can be disabled with glDisableClientState(GL_VERTEX_ARRAY);

Pixel Arrays

- Pixmaps -- rectangular arrays of colour values.
- If only have colour-depth 1 then called a *bitmap*.
- In OpenGL can draw this using:

  glBitmap(width, height, x0, y0, xOffset, yOffset, bitShape); //for bitmaps

  glDrawPixels(width, height, dataFormat, dataType, pixMap);// for pixmaps
- Note data format can be things like GL_RGB. Datatype might be GL_INT

More Pixel Arrays

Example code fragment:

```
GLubyte bitShape[20] = {
    0x1c, 0x00, 0x1c, 0x00, 0x1c, 0x00, 0x1c, 0x00,
      0x1c, 0x00,
    0xff, 0x80, 0x7f, 0x00, 0x 3e, 0x00, 0x1c, 0x00,
      0x08, 0x00};
glPixelStorei(GL_UNPACK_ALIGNMENT,1);
glRasterPos2i(30,40);
glBitmap(9, 10, 0.0, 0.0, 20.0, 15.0, bitShape);
```

More on Pixmaps

- If using a buffer can specify buffer to draw to using

  glDrawBuffer(buffer); //GL_BACK
- Can read a group of pixels using

  glReadPixel(xmin,ynim, width, height,
      dataformat, dataType, array);
- Can set buffer to read to with glReadBuffer(buffer)