Scan Conversion: Polygons

Lines (last lecture)

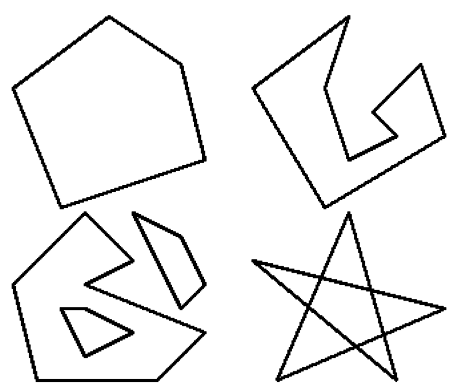
Polygons

- Types of polygons
- Inside tests
- Jordan Curve Theorem
- Singularities
- Scanline algorithms

Types of Polygons

Types

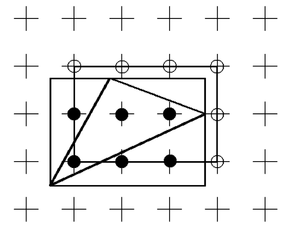
- Triangles
- Trapezoids
- Quadrilaterals
- Convex
- Concave
- Self-intersecting
- General
  - Multiple loops
  - Holes
- Spline-gons



Two approaches:
1. Generalize scan conversion
2. Tessellate into triangles

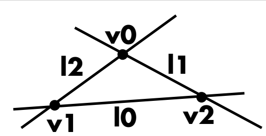
Triangle Tiler

```
tile3( vert v[3] )
{
  int x, y;
  bbox b;

  bound3(v,&b);
  for( y=b.ymin; y<b.ymax, y++ )
    for( x=b.xmin; x<b.xmax, x++ )
      if( inside3(v,x,y) )
        fragment(x,y);
}
```

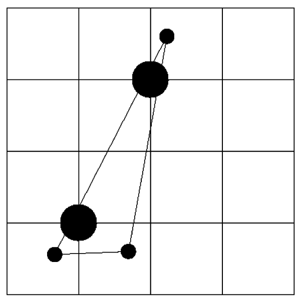
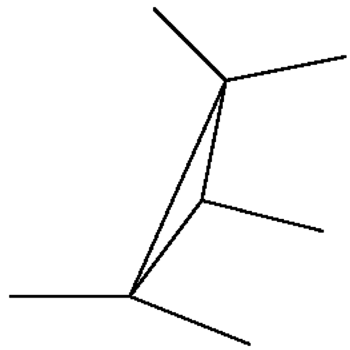


Incremental Triangle Tiler

```
edge l0, l1, l2; value e0, e1, e2;

mkedge(&v[0],&v[1],&l2);
mkedge(&v[1],&v[2],&l0);
mkedge(&v[2],&v[0],&l1);

e0 = l0.a * b.xmin + l0.b * b.ymin + l0.c;
e1 = l1.a * b.xmin + l1.b * b.ymin + l1.c;
e2 = l2.a * b.xmin + l2.b * b.ymin + l2.c;

for( y=b.ymin; y<b.ymax, y++ ) {
  for( x=b.xmin; x<b.xmax, x++ ) {
    if( e0<=0 && e1<=0 && e2<=0 ) fragment(x,y);
    e0 += l0.a; e1 += l1.a; e2 += l2.a;
  }
  e0 += l0.a * (b.xmin - b.xmax) + l0.b; ...
}
```



Gaps and T-Vertices

Ok! **Bad!**

CS248 Fall 98 Lecture 9

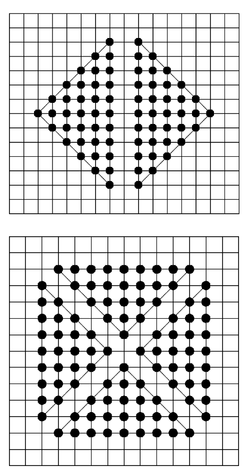
Singularities

Double hits (e <= 0)

- **Wasted effort**
- **Transparency**

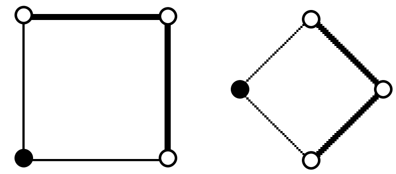
Gaps (e < 0)

- **Missing pixels**



CS248 Fall 98 Lecture 9

Shadow Test

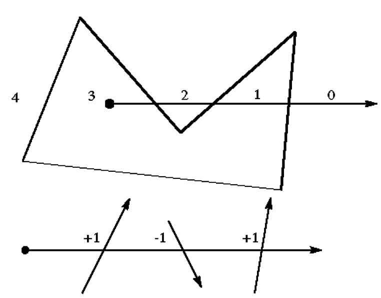


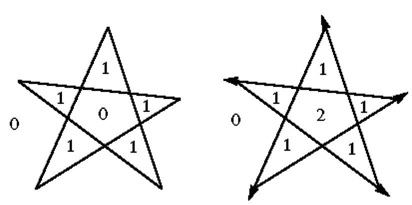
```
int shadow( value a, value b ) {
  return (a>0) || (a==0 && b > 0);
}
int inside( value e, value a, value b ) {
  return (e == 0) ? !shadow(a,b) : (e < 0);
}
```


Jordan Curve Theorem

Two definitions of inside

- **Even-Odd parity**
- **Winding number**





Parity **Winding Number**

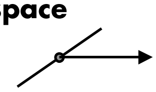
Singularities

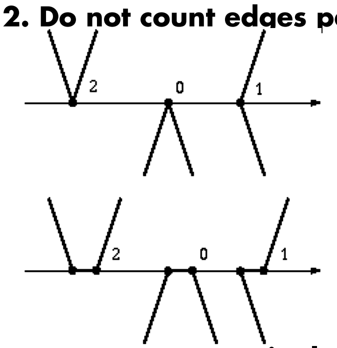
To cross: 1 vertex above and 1 below horizontal line

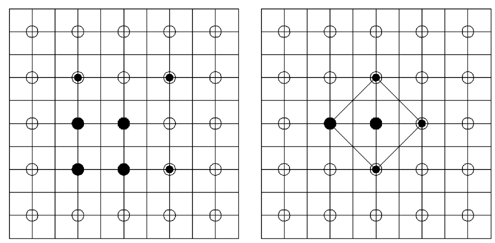
Resolve singularities

1.Vertices on horizontal line are part of lower half space

2. Do not count edges passing through point

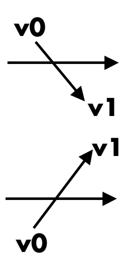




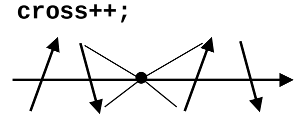


Equivalent to half-open intervals and shadow test

Point in Polygon




```
int inside( vert *v, int n, int x, int y ) {
  int cross = 0; int x0, y0, x1, y1;

  x1 = v[n-1].x - x; y1 = v[n-1].y - y;
  while( n-- ) {
    x0 = x1;       y0 = y1;
    x1 = v->x - x; y1 = v->y - y;
    if( y0 > 0 ) {
      if( y1 <= 0 )
        if( x1*y0 > y1*x0 ) cross++;
    }
    else {
      if( y1 > 0 )
        if( x0*y1 > y0*x1 ) cross++;
    }
    v++;
  }
  return cross & 1;
```



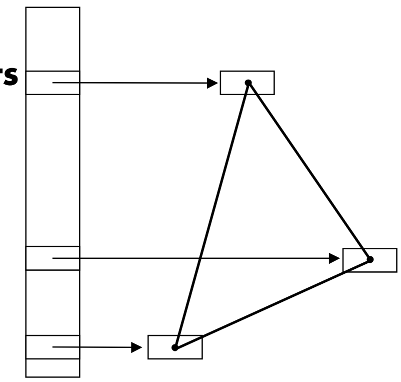
Scanline Algorithm

Incrementalize Jordan Test via program transformation

1. Priority queue of vertex events

For each polygon vertex

Bucket sort by y value

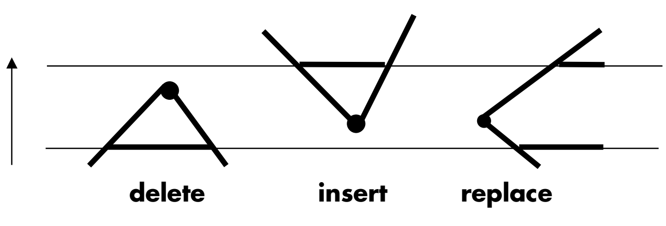


Scanline Algorithm

2. Sweep

For each scanline

A. Maintain *active edge list* (for crossings count)

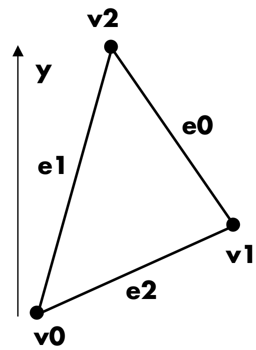


B. Increment x intercepts, Sort by x intercepts

C. Output span between left and right edge

Triangle Tiler

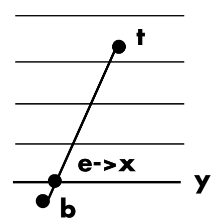
```
scany( vert v[3]) {

  ybot = ceil(v[0].y);
  ymid = ceil(v[1].y);
  ytop = ceil(v[2].y);

  differencey(&v[0],&v[2],&e1,ybot); l=e1;
  differencey(&v[0],&v[1],&e2,ybot); r=e2;
  differencey(&v[1],&v[2],&e0,ymid);

  for( y=ybot; y<ytop; y++ ) {
      if( y >= ymid ) r = e0;
      scanx(&l,&r,y);
      l.x += l.dxdy;
      r.x += r.dxdy;
  }
}
```



Assume:
1. v's sorted
2. single l edge

Triangle Tiler

```
differencey(vert *b, vert *t, edge *e, int y)
{
  e->dxdy = (t->x - b->x) / (t->y - b->y);
  e->x    = b->x + (y - b->y) * e->dxdy;
}
```




```
scanx( edge *l, edge *r, int y)
{
  lx = ceil(l->x);
  rx = ceil(r->x);

  for( x=lx; x<rx; x++ )
      fragment(x,y);
}
```