Lecture 8 of 41

Scan Conversion 2 of 2:
Circles/Ellipses and Polygons

William H. Hsu
Department of Computing and Information Sciences, KSU

Readings:
Today: Sections 2.4, 2.5 esp. 2.5.4, 3.1.6, Eberly
Next class: Sections 2.5, 2.6.1-2.6.2, 4.3.2, 20.2, Eberly

Excerpts from Van Dam notes, Brown CS123
Scan converting circles/ellipses (starting from 19 in Fall, 2010 notes)
Polygons (Shapes 2-4)
Triangle meshes (Shapes 13-14)
Scan line interpolation (Polygons 7; Shading 14, 2005 – 2009 notes)

Last Time: Intro to Clipping and Culling
Clipping: Cohen-Sutherland, Cyrus-Beck / Liang-Barsky
Visibility Culling: view frustum, back face, occlusion
Today: Scan Conversion, Concluded
Circles and ellipses
Polygons

Where We Are

Drawing Circles, Version 3

Symmetry: If \((x_0, y_0 + b)\) is on circle
also \((x_0, y_0 - b)\) and \((x_0 + b, y_0)\), \((x_0 - b, y_0)\), hence 8-way symmetry.
Reduce the problem to finding the points for 1/8 of the circle.

Using The Symmetry

Scan top right 1/8 of circle of radius \(R\)
Circle starts at \((x_0, y_0 + R)\)
Let’s use another incremental algorithm with decision variable evaluated at midpoint

Adapted from slides © 1997 – 2010 van Dam et al., Brown University
Incremental Algorithm [1]: Sketch

\[ x = x_0, \ y = y_0; \ R. \ \text{Func}(x, y); \]

for \( x = x_0 + 1 \) \{ \( x = x_0 \) \}

1. If decision var \( d \leq 0 \) \{ update decision variable \}
   1. north west \( \rightarrow \) update decision variable
   2. If \( y < y_0 + \frac{1}{2} \) \{ update decision variable \}
   3. If \( y = y_0 + \frac{1}{2} \) \{ update decision variable \}
   4. If \( y > y_0 + \frac{1}{2} \) \{ update decision variable \}
   5. Note: can replace all occurrences of \( x_0, y_0 \) with 0.

Decision Variable

- \( f(x, y) = x^2 + y^2 - R^2 \)
- \( f(x, y) \) is zero on circle, negative inside, positive outside
- If we are at pixel \((x, y)\) examine \((x + 1, y)\) and \((x + 1, y - 1)\)
- Compute \( f \) at the midpoint

Incremental Algorithm [2]: Computations needed

- Decision variable
  - negative if we move E, positive if we move SE (or vice versa)
- Follow line strategy: Use implicit equation of circle
  - \( f(x, y) = x^2 + y^2 - R^2 = 0 \)
  - \( f(x, y) \) is zero on circle, negative inside, positive outside
- If we are at pixel \((x, y)\) examine \((x + 1, y)\) and \((x + 1, y - 1)\)
- Compute \( f \) at the midpoint

Decision Variable

- \( f(x, y) = x^2 + y^2 - R^2 \) at the point:
  - \( f_x = 2x \)
  - \( f_y = 2y \)

We are asking: Is \( f(x + 1, y - 1) = f(x + 1, y) - \frac{1}{2} \)?

positive or negative? (It is zero on circle)

- If negative, midpoint inside circle, choose E.
- Vertical distance to the circle is less at \((x + 1, y)\) than at \((x + 1, y - 1)\).
- If positive, opposite is true, choose SE

Right Decision Variable?

- Decision based on vertical distance
- Ok for lines, since \( d \) and \( d_{new} \) are proportional
- For circles, not true:
  - \( d((x + 1, y), \text{Circ}) = \sqrt{(x + 1)^2 + y^2 - R} \)
  - \( d((x + 1, y - 1), \text{Circ}) = \sqrt{(x + 1)^2 + (y - 1)^2 - R} \)

Which \( d \) is closer to zero? (i.e., which of the two values above is closer to \( R \):

\( \sqrt{(x + 1)^2 + y^2} \) or \( \sqrt{(x + 1)^2 + (y - 1)^2} \)

Alternate Phrasing [1]

- We could ask instead: 
  - “Is \((x + 1)^2 + y^2 \) or \((x + 1)^2 + (y - 1)^2\) closer to \( R^2\)?”

The two values in equation above differ by

\[ ([x + 1]^2 + y^2] - ([x + 1]^2 + (y - 1)^2]) = 2y - 1 \]

Alternate Phrasing [2]

- The second value, which is always less, is closer if its difference from \( R^2 \) is less than \( \frac{1}{2} (2y - 1) \)

i.e., if

\[ R^2 - ([x + 1]^2 + (y - 1)^2] < \frac{1}{2} (2y - 1) \]

then

\[ 0 < y - \frac{1}{2} + (x + 1)^2 + (y - 1)^2 - R^2 \]

\[ 0 < (x + 1)^2 + y^2 - y - \frac{1}{2} - R^2 \]

\[ 0 < (x + 1)^2 + y^2 - y - \frac{1}{2} - R^2 \]

\[ 0 < (x + 1)^2 + (y - 1)^2 - \frac{1}{2} - R^2 \]
Alternate Phrasing [3]

- The radial distance decision is whether
  \[ d_1 = (x + 1)^2 + \left(y - \frac{1}{2}\right)^2 - \frac{4}{3} R^2 \]
  is positive or negative.

- The vertical distance decision is whether
  \[ d_2 = (x + 3)^2 + (y - \frac{1}{2})^2 - R^2 \]
  is positive or negative; \(d_1\) and \(d_2\) are \(\frac{1}{4}\) apart.

- The integer \(d_2\) is positive only if \(d_2 + \frac{1}{4}\) is positive (except special case where \(d_2 = 0\); remember you’re using integers).

Incremental Algorithm Revisited [1]

- How to compute the value of
  \[ f(x,y) = (x + 1)^2 + \left(y - \frac{1}{2}\right)^2 - R^2 \]
  at successive points? (vertical distance approach)

- Answer: Note that \(f(x + 1, y) = f(x, y)\)
  \[ \Delta y = 2x + 3 \]
  and that
  \[ f(x + 1, y - 1) - f(x, y) \]
  is just
  \[ \Delta y(x, y) = 2x + 3 - 2y + 2 \]

Incremental Algorithm Revisited [2]

- If we move E, update by adding \(2x + 3\)
- If we move SE, update by adding \(2x + 3 - 2y + 2\)
- Forward differences of a 1st degree polynomial are constants and those of a 2nd degree polynomial are 1st degree polynomials
- This “first order forward difference”, like a partial derivative, is one degree lower.

Second Differences [1]

- The function \(\Delta y(x, y) = 2x + 3\) is linear; hence amenable to incremental computation:
  \[ \Delta y(x + 1, y) - \Delta y(x, y) = 2 \]
  \[ \Delta y(x + 1, y - 1) - \Delta y(x, y) = 4 \]

Similarly
  \[ \Delta x(x + 1, y) - \Delta x(x, y) = 2 \]
  \[ \Delta x(x + 1, y - 1) - \Delta x(x, y) = 4 \]

Second Differences [2]

- For any step, can compute new \(\Delta y(x, y)\) from old \(\Delta y(x, y)\) by adding appropriate second constant increment – update delta terms as we move.
  - This is also true of \(\Delta x(x, y)\).
- Having drawn pixel \((a, b)\), decide location of new pixel at \((a + 1, b)\) or \((a + 1, b - 1)\), using previously computed \(\Delta y(a, b)\).
- Having drawn new pixel, must update \(\Delta y(a, b)\) for next iteration; need to
  - find either \(\Delta y(a + 1, b)\) or \(\Delta y(a + 1, b - 1)\) depending on pixel choice
  - Must add \(\Delta y(x, y)\) or \(\Delta x(x, y)\) to \(\Delta y(x, y)\).
- So we:
  - Look at \(x\) to decide which to draw next, update \(x\) and \(y\)
  - Update \(\Delta y(x, y)\) by \(\Delta y(x, y)\):
    - Update each of \(\Delta y(a, b)\) and \(\Delta y(x, y)\) for future use.
  - Draw pixel (x,y).

Midpoint Eighth Circle Algorithm

```c
MBC(r) = 1/Bth of circle with radius r */
int deltaxE = 2; deltaxSE = 2; /* Update deltax */
int deltaxNE = 4; deltaxNW = 4; /* Update deltax */

void drawCircle(int x, int y, int radius) {
    int x = 0; y = radius;
    int dx = deltaxE; dy = deltaxNE; /* Delta x and y */

    while (x < y) {
        if (dy > dy) { /* More east */
            deltaxE = deltaxE + deltaxSE; /* Update deltax */
            if (dy > dy) { /* More SE */
                deltaxE = deltaxE + deltaxNE; /* Update deltax */
                dx = dx + dx; /* Update dx */
            }
        } else { /* More NW */
            deltaxNE = deltaxNE + deltaxNW; /* Update deltax */
            if (dy > dy) { /* More NE */
                deltaxNE = deltaxNE + deltaxNE; /* Update deltax */
                dx = dx + dx; /* Update dx */
            }
        }
        drawPixel(x, y);
        y = y + dy; /* Update y */
    }
}
```
Analysis

- Uses floats!
- 1 test, 3 or 4 additions per pixel
- Initialization can be improved
- Multiply everything by 4: No Floats!
- Makes the components even, but signs of decision variable remain same

Questions
- Are we getting all pixels whose distance from the circle is less than t? Why is p > 0 the right stepping criterion? What if d were an ellipse?

Other Scan Conversion Problems

- Aligned Ellipses
- Non-integer primitives
- General conics
- Patterned primitives

Aligned Ellipses

- Equation is \( \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \)
- Computation of \( \Delta x \) and \( \Delta y \) is similar
- Only 4-fold symmetry
- When do we stop stepping horizontally and switch to vertical?

Direction-Changing Criterion [1]

- When absolute value of slope of ellipse is more than 1:
  - How do you check this? At a point \((x, y)\) for which \(f(x, y) = 0\), a vector perpendicular to the level set is \(f(x, y)\) which is \([\frac{df}{dx}(x, y), \frac{df}{dy}(x, y)]\)
  - This vector points more right than up when \(\frac{df}{dx}(x, y) > \frac{df}{dy}(x, y)\)

Direction-Changing Criterion [2]

- In our case \(\frac{df}{dx}(x, y) = 2ax \) and \(\frac{df}{dy}(x, y) = 2by\)

so we check for
\[
2ax^2 - 2by^2 > 0 \\
a^2x^2 - b^2y^2 > 0
\]

- This, too, can be computed incrementally

Problems with Aligned Ellipses

- New in ENE octant, not ESE octant
- This problem is an artifact of aliasing, remember filter?
Patterned Lines

- Patterned line from P to Q is not the same as patterned line from Q to P.

Geometric vs. Cosmetic

- Patterns can be geometric or cosmetic.
- Cosmetic texture applied after geometric transformations.
- Geometric patterns subject to transformations.

Non-Integer Primitives & General Conics

- Non-Integer Primitives:
  - Initialization is harder
  - Endpoints are hard, too
  - Making Line \((P, Q)\) and Line \((Q, R)\) join properly is a good test
  - Symmetry is lost

- General Conics:
  - Very hard—the octant-changing test is tougher, the difference computations are tougher, etc.
  - Do it only if you have to
  - Note that drawing gray-scale conics is easier than drawing BW conics

2-D Object Definition [1]

- Lines and polylines: lines drawn between ordered points
  - A closed polygon is a polygon, a simple polygon has no self-intersections
  - Cones and concave polygons:
    - Cone: For every pair of points inside the polygon, the line between them is entirely inside the polygon.
    - Convex: For some pair of points inside the polygon, the line between them is not entirely inside the polygon. Not Convex.

2-D Object Definition [2]

- Special Polygons:
  - Triangles
  - Squares
  - Rectangles
  - Circles

- A circle can be approximated by a polygon with many sides.

Triangle Meshes

- Most common representation of shape in three dimensions
- All vertices of triangle are guaranteed to lie in one plane (not true for quadrilaterals or other polygons)
- Uniformity makes it easy to perform mesh operations such as subdivision, simplification, transformation etc.
- Many different ways to represent triangular meshes

- See chapters 9 and 12 in book, en.wikipedia.org/wiki/polygon_mesh
- Mesh transformation and deformation
- Procedural generation techniques (upcoming lab on simulating terrain)
Triangular Mesh Representation

- Vertex and face tables, analogous to 2D vertex and edge tables.
- Each vertex listed once, triangles listed as ordered triplets of indices into the vertex table.
- Edges often occur twice.
- It's often useful to store associated indices with vertices (i.e., computing normals, vertex normal average of surrounding face normals).
- Vertices listed in counter-clockwise order in face table.
- No longer just because of convention: CROP index differentiates front and back of face.

Brown University CS123
Cohen
Fall 2010

Adapted from slides © 1997 – 2010 van Dam et al., Brown University
Reused with permission.

General Polygons [1]: Scan Line Interpolation

1. Interpolate Value Along Polygon Edges to Get \( I_a, I_b \)
2. Interpolate Value Along Scan Lines to Get \( I_s \)

\[
\begin{align*}
I_s &= \frac{x_s - x_0}{x_1 - x_0} I_a + \frac{x_1 - x_s}{x_1 - x_0} I_b \\
I_s &= \frac{x_s - x_0}{y_1 - y_0} I_a + \frac{y_1 - x_s}{y_1 - y_0} I_b
\end{align*}
\]

Brown University CS123
Liang
Fall 2010

Adapted from slides © 1997 – 2010 van Dam et al., Brown University
Reused with permission.

General Polygons [2]: Texture Mapping Preview

- (u,v,w) texture coordinates are pre-calculated and specified per vertex.
- Vertices may have different texture coordinates for different faces.
- Texture coordinates are linearly interpolated across polygon.

Brown University CS123
Brown University
Fall 2010

Adapted from slides © 1997 – 2010 van Dam et al., Brown University
Reused with permission.

General Polygons [3]: Continuity and Scan Line Interpolation

- what’s the difference between these two solutions? Under which circumstances is the right one “better”?

Brown University CS123
Barrett
Fall 2010

Adapted from slides © 1997 – 2010 van Dam et al., Brown University
Reused with permission.

Summary

- Last Time: Clipping and Culling
  - What parts of scene to clip: edges vs. polygons of model
  - What parts of viewport to clip against: clip faces vs. clip edges
  - Cohen-Sutherland clipping: outcodes, simultaneous equations
  - Liang-Barsky / Cyrus-Beck clipping: parametric equations
  - Visibility cutting: view frustum, back face, occlusion
- Today: Scan Conversion, Concluded
  - Circles and ellipses
  - Polygons: scan line interpolation (for flat/constant shading)
  - Later: Gouraud & Phong shading, z-buffering, texture mapping
- Excerpts from Van Dam notes, Brown CS123
  - Scan converting circles/ellipses
  - Polygons
  - Triangle meshes
  - Scan line interpolation

Terminology

- Scan Conversion (aka Rasterization)
  - Given: geometric object (e.g., circle, ellipse, projected polygon)
  - Decide: what pixels to light (turn on; later, color/shade)
  - Basis: what part of pixels crossed by object
- Issues (Reasons why Scan Conversion is Nontrivial Problem)
  - Aliasing (e.g., jaggies) – discontinuities in lines
  - Cracks: discontinuities in “polygon” mesh
- Drawing Circles & Ellipses
  - Incremental algorithm – uses rounding, floating point arithmetic
  - Forward differences – precalculated amounts to add to running total
  - Decision variable – value whose sign indicates which pixel is next
- Drawing Polygons
  - Texture mapping – finding pixels of image (texture) to put in polygon
  - Scan line interpolation – procedure for filling in closed curves