Bresenham Circles

CS5600 Intro to Computer Graphics
From Rich Riesenfeld
Spring 2013

More Raster Line Issues

- Fat lines with multiple pixel width
- Symmetric lines
- End point geometry – how should it look?
- Generating curves, e.g., circles, etc.
- Jaggies, staircase effect, aliasing...

Generating Circles

Exploit 8-Point Symmetry

Once More: 8-Pt Symmetry

Only 1 Octant Needed

We will generate 2nd Octant
Generating pt \((x,y)\) gives
the following 8 pts by symmetry:
\{(x,y), (-x,y), (-x,-y), (x,-y),
(y,x), (-y,x), (-y,-x), (y,-x)\}

2\textsuperscript{nd} Octant Is a Good Arc

- It is a function in this domain
  - single-valued
  - no vertical tangents: \(|\text{slope}| \leq 1\)
- Lends itself to Bresenham
  - only need consider \(E\) or \(SE\)

Implicit Eq's for Circle

- Let \(F(x,y) = x^2 + y^2 - r^2\)
- For \((x,y)\) on the circle, \(F(x,y) = 0\)
- So, \(F(x,y) > 0 \Rightarrow (x,y) \text{ Outside}\)
- And, \(F(x,y) < 0 \Rightarrow (x,y) \text{ Inside}\)

Choose \(E\) or \(SE\)

- Function is \(x^2 + y^2 - r^2 = 0\)
- So, \(F(M) \geq 0 \Rightarrow SE\)
- And, \(F(M) < 0 \Rightarrow E\)
**Decision Variable $d$**

Again, we let,

$$d = F(M)$$

---

**Look at Case 1: $E$**

Let ideal curve:

- $E$
- $M$
- $SE$

---

Consider $d_{old}$:

$$d_{old} < 0 \implies E$$

Then,

$$d_{old} = F(x_p + 1, y_p - \frac{1}{2})$$

$$= (x_p + 1)^2 + (y_p - \frac{1}{2})^2 - r^2$$

---

Consider $d_{new}$:

$$d_{new} = d_{old} + (2x_p + 3)$$

Since,

$$d_{new} - d_{old}$$

$$= (x_p + 2)^2 - (x_p + 1)^2 = (x_p^2 + 4x_p + 4) - (x_p^2 + 2x_p + 1)$$

$$= 2x_p + 3$$

---

**Extra Case:**

If $d_{old} < 0 \implies E$

- $d_{new} = d_{old} + \Delta E$

where,

$$\Delta E = 2x_p + 3$$
Look at Case 2: $SE$

$$d_{old} \geq 0 \implies SE$$

$$d_{new} - d_{old} =$$

$$= (x_p + 2)^2 + (y_p - \frac{3}{2})^2 - r^2 - [(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - r^2]$$

$$= (2x_p + 3) + y_p^2 - 3y_p + \frac{9}{4} - \left[ y_p^2 - y_p + \frac{1}{4} \right]$$

Because, ..., straightforward manipulation

$$d_{old} \geq 0 \implies SE$$

$$d_{new} = F(x_p + 2, y_p - \frac{3}{2})$$

$$= (x_p + 2)^2 + (y_p - \frac{3}{2})^2 - r^2$$

$$d_{new} = d_{old} + (2x_p - 2y_p + 5)$$

$$d_{old} \geq 0 \implies SE$$

$$d_{new} - d_{old} =$$

$$= (x_p + 2)^2 + (y_p - \frac{3}{2})^2 - r^2 - [(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - r^2]$$

$$= (2x_p + 3) + y_p^2 - 3y_p + \frac{9}{4} - \left[ y_p^2 - y_p + \frac{1}{4} \right]$$

$$d_{old} \geq 0 \implies SE$$

$$d_{new} - d_{old} =$$

$$= (2x_p + 3) + (-3y_p + \frac{9}{4}) - (-y_p + \frac{1}{4})$$

From $\Delta E$ calculation

From $new$ y-coordinate

From $old$ y-coordinate
\[ d_{old} \geq 0 \Rightarrow SE \]

I.e.,

\[ d_{new} = d_{old} + (2x_p - 2y_p + 5) \]

\[ = d_{old} + \Delta_SE \]

\[ \Delta_SE = 2x_p - 2y_p + 5 \]

Note: \( \Delta \)'s Not Constant

\( \Delta_E \) and \( \Delta_SE \)

depend on values of \( x_p \) and \( y_p \)

Summary

• \( \Delta \)'s are no longer constant over entire line
• Algorithm structure is exactly the same
• Major difference from the line algorithm
  – \( \Delta \) is re-evaluated at each step
  – Requires real arithmetic

Initial Condition

• Let \( r \) be an integer. Start at \((0, r)\)
• Next midpoint \( M \) lies at \((l, r - \frac{1}{2})\)
• So, \( F(l, r - \frac{1}{2}) = 1 + (r^2 - r - \frac{1}{4})r^2 \)

\[ = \frac{5}{4} - r \]

Ellipses

• Evaluation is analogous
• Structure is same
• Have to work out the \( \Delta \)'s

Getting to Integers

• Note the previous algorithm involves real arithmetic
• Can we modify the algorithm to use integer arithmetic?
**Integer Circle Algorithm**

- Define a shift decision variable
  \[ h = d - \frac{1}{4} \]
- In the code, plug in \( d = h + \frac{1}{4} \)

**Integer Circle Algorithm**

- Now, the initialization is \( h = 1 - r \)
- So the initial value becomes
  \[ F(1, r - \frac{1}{2}) - \frac{1}{4} = \frac{5}{4} - r - \frac{1}{4} \]
  \[ = 1 - r \]

**Integer Circle Algorithm**

- Then, \( d < 0 \) becomes \( h < -\frac{1}{4} \)
- Since \( h \) an integer

\[ h < -\frac{1}{4} \iff h < 0 \]

**End of Bresenham Circles**

**Another Digital Line Issue**

- Clipping Bresenham lines
- The integer slope is not the true slope
- Have to be careful
- More issues to follow
Line Clipping Problem

\[ x = x_{\text{min}} \quad \text{Clipping Rectangle} \]

\[ (x_0, y_0) \]

\[ (x_1, y_1) \]

\[ x = x_{\text{max}} \]

\[ y = y_{\text{min}} \]

Clipped Line

\[ (x_0', y_0') \]

\[ (x_1', y_1') \]

\[ y = y_{\text{max}} \]

\[ x = x_{\text{min}} \]

\[ x = x_{\text{max}} \]

Drawing Clipped Lines

\[ (x_0, y_0) \]

\[ (x_1, y_1) \]

Clipped Line Has Different Slope!

\[ m = \frac{1}{2} \]

\[ m = \frac{3}{4} \]

Pick Right Slope to Reproduce Original Line Segment

Zoom of previous situation

Pick Right Slope to Reproduce Original Line Segment

Zoom of previous situation
Clipping Against $x = x_{\text{min}}$

- Situation is complicated
- Multiple pixels involved at $(y = y_{\text{min}})$
- Want all of those pixels as “in”
- Analytic $\cap$, rounding $x$ gives $A$
- We want point $B$

Clipping Against $y = y_{\text{min}}$

- Use $\text{Line} \cap y = y_{\text{min}} - \frac{1}{2}$
- Round up to nearest integer $x$
- This yields point $B$, the desired result

Jaggies-Manifestation of Aliasing

Added resolution helps, but does not directly address underlying issue of aliasing

Jaggies and Aliasing

- To represent a line with discrete pixel values is to sample finitely a continuous function
- Jaggies are visual manifestation, artifacts, resulting from information loss
- The term aliasing is a complicated, unintuitive phenomenon which will be defined later
Jaggies and Aliasing

- Doubling resolution in x and y reduces the effect of the problem, but does not fix it
- Doubling resolution costs 4 times memory, memory bandwidth and scan conversion time!

Anti-aliasing

Pixel intensity (darkness, in this case) is proportional to area covered by line

Anti-aliasing

- Set each pixel’s intensity value proportional to its area of overlap (i.e. sub-area) covered by primitive
- Not more than 1 pixel/column for lines with \( 0 < \text{slope} < 1 \)

Gupta-Sproull Algorithm -1

- Standard Bresenham chooses \( E \) or \( NE \)
- Incrementally compute distance \( D \) from chosen pixel to center of line
- Vary pixel intensity by value of \( D \)
- Do this for line above and below

Gupta-Sproull Algorithm -2

- Use coarse (4-bit, say) lookup table for intensity: \( \text{Filter} (D, t) \)
- Note, \( \text{Filter} \) value depends only on \( D \) and \( t \), not the slope of line! (Very clever)
- For line-width \( t = 1 \) geometry and associated calculations greatly simplify
Cone Filter for Weighted Area Sampling

Observations

- Lines are complicated
- Many aspects to consider
- We omitted many
- What about intensity of $y = x \ vs \ y = 0$?

The End

Bresenham Circles