Today

- 2D Scan Conversion
  - Drawing Lines
  - Drawing Curves
  - Filled Polygons
  - Filling Algorithms
Drawing a Line

- Basically, it's easy... but for the details
- Lines are a basic primitive that needs to be done well...

Drawing a Line

$p_1 = (x_1, y_1)$

$p_2 = (x_2, y_2)$
Some things to consider

- How thick are lines?
- How should they join up?
- Which pixels are the right ones?

For example:
Drawing a Line

\[ y = m \cdot x + b, x \in [x_1, x_2] \]

\[ m = \frac{y_2 - y_1}{x_2 - x_1} \]

\[ b = y_1 - m \cdot x_1 \]
Drawing a Line

\[ \Delta x = 1 \]
\[ \Delta y = m \cdot \Delta x \]

x = x1
y = y1
while (x <= x2)
    plot(x, y)
    x++
    y += Dy
Drawing a Line

Δx = 1
Δy = m \cdot Δx

After rounding
Drawing a Line

\[ \Delta x = 1 \]
\[ \Delta y = m \cdot \Delta x \]
\[ y += \Delta y \]

Accumulation of roundoff errors

How slow is float-to-int conversion?

\[ |m| \leq 1 \]
\[ |m| > 1 \]
void drawLine(int x1, x2, int y1, y2)

float m = (y2 - y1) / (x2 - x1)
int x = x1
float y = y1

while (x <= x2)
    setPixel(x, round(y), PIXEL_ON)
    x += 1
    y += m
void drawLine-Error1(int x1, x2, int y1, y2)

float m = \( \frac{y2-y1}{x2-x1} \)
int x = x1
float y = y1

while (x <= x2)
    setPixel(x, round(y), PIXEL_ON)
    x += 1
    y += m

void drawLine-Error2(int x1, x2, int y1, y2)

float m = \( \frac{y2-y1}{x2-x1} \)
int x = x1
int y = y1
float e = 0.0

while (x <= x2)
    setPixel(x, y, PIXEL_ON)
    x += 1
    e += m
    if (e >= 0.5)
        y += 1
        e -= 1.0

Tuesday, October 7, 2008
void drawLine>Error2(int x1,x2, int y1,y2)
    float m = float(y2-y1)/(x2-x1)
    int x = x1
    int y = y1
    float e = 0.0

    while (x <= x2)
        setPixel(x,y,PIXEL_ON)
        x += 1
        e += m
        if (e >= 0.5)
            y+=1
            e-=1.0

void drawLine>Error3(int x1,x2, int y1,y2)
    int x = x1
    int y = y1
    float e = -0.5

    while (x <= x2)
        setPixel(x,y,PIXEL_ON)
        x += 1
        e += float(y2-y1)/(x2-x1)
        if (e >= 0.0)
            y+=1
            e-=1.0

No more rounding
void drawLine-Error4(int x1,x2, int y1,y2)

int x = x1
int y = y1
float e = -0.5*(x2-x1)  // was -0.5

while (x <= x2)
    setPixel(x,y,PIXEL_ON)
    x += 1
    e += y2-y1              // was /(x2-x1)
    if (e >= 0.0)           // no change
        y+=1
    e-=(x2-x1)              // was 1.0

void drawLine-Error5(int x1,x2, int y1,y2)

int x = x1
int y = y1
int e = -(x2-x1)  // removed *0.5

while (x <= x2)
    setPixel(x,y,PIXEL_ON)
    x += 1
    e += 2*(y2-y1)         // added 2*
    if (e >= 0.0)          // no change
        y+=1
    e-=(x2-x1)             // added 2*
Drawing a Line

```c
void drawLine-Bresenham(int x1, int y1, int x2, int y2)
{
    int x = x1
    int y = y1
    int e = -(x2-x1)
    while (x <= x2)
    {
        setPixel(x,y,PIXEL_ON)
        x += 1
        e += 2*(y2-y1)
        if (e >= 0.0)
        {
            y+=1
            e-=2*(x2-x1)
        }
    }
}
```

| Faster |
| Not wrong |

\[
|m| \leq 1 \quad x_1 \leq x_2
\]

19

Drawing Curves

\[
y = f(x)
\]

Only one value of \(y\) for each value of \(x\)...
Drawing Curves

- Parametric curves
  - Both $x$ and $y$ are a function of some third parameter

\[
x = f(u) \\
y = f(u) \\
x = f(u) \\
u \in [u_0 \ldots u_1]
\]
Drawing Curves

- Draw curves by drawing line segments
  - Must take care in computing end points for lines
  - How long should each line segment be?

$$x = f(u) \quad u \in [u_0 \ldots u_1]$$
Drawing Curves

- Midpoint-test subdivision

\[ |f(u_{mid}) - l(0.5)| \]
Drawing Curves

- Midpoint-test subdivision

\[ |f(u_{\text{mid}}) - l(0.5)| \]

Not perfect

- We need more information for a guarantee...

\[ |f'(u_{\text{mid}}) - l'(0.5)| \]
Filled Polygons

What happens at these locations?

Filled Polygons

If we count ONCE...
Filled Polygons

If we count TWICE...

Treat \((\text{scan } y = \text{vertex } y)\) as \((\text{scan } y > \text{vertex } y)\)
Filled Polygons

- “Equality Removal” applies to all vertices
- Both $x$ and $y$ coordinates

Filled Polygons

- Final result:
Filled Polygons

- Who does this pixel belong to?

Tuesday, October 7, 2008
Drawing a Line

- How thick?
  -
  -
  -

- Ends?
  - Butt
  - Round
  - Square

Drawing a Line

- Joining?
  - Ugly
  - Bevel
  - Round
  - Miter
Inside/Outside Testing

The Polygon

Non-exterior

Non-zero winding

Parity

Optimize for Triangles

- Split triangle into two parts
  - Two edges per part
  - Y-span is monotonic
- For each row
  - Interpolate span
- Interpolate barycentric coordinates
Flood Fill

Flood Fill

Start Position