Binghamton University

EngiNet™

State University of New York

Thomas J. Watson

School of Engineering
and Applied Science

EngiNet™

WARNING

CS 560

Computer Graphics

Professor Richard Eckert

Lecture # 8

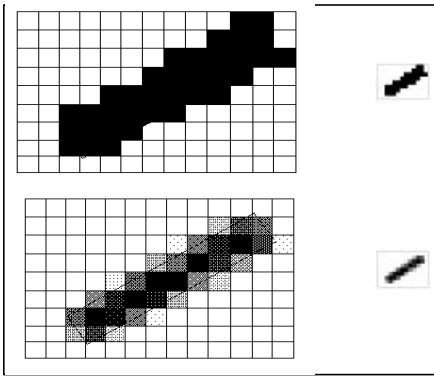
February 14, 2001

Scan Conversion Algorithms for 2D Output Primitives

- Straight Lines
- Polygons
- Circles
- Ellipses and Other 2-D Curves
- Text (Characters)

Aliasing (Jaggies)

- Inherent in Raster Scan systems
- Anti-aliasing technique for grayscale:
  - Consider broad line covering many pixels
  - Border pixels
    - Set intensity to % of pixel inside line
    - Produces blurring
    - Looks less jagged



Polyline Algorithm

```
Polyline (POINT *p, int n)
{
int xo, yo, xn, yn;
if (n==0) return;
xo=p[0].x; yo=p[0].y;
if (n==1) {SetPixel(xo, yo); return;}
for (i=1; i<n; i++)
    {xn=p[i].x; yn=p[i].y;
    Line(xo,yo,xn,yn);
    xo=xn; yo=yn;}
}
```

Calling the Polyline Algorithm

```
POINT pt[3];
pt[0].x=50; pt[0].y=10;
pt[1].x=250; pt[1].y=50;
pt[2].x=125; pt[2].y=130;
Polyline(pt,3);
```

Scan Converting Circles

Given:

Center: (h,k)

Radius: r

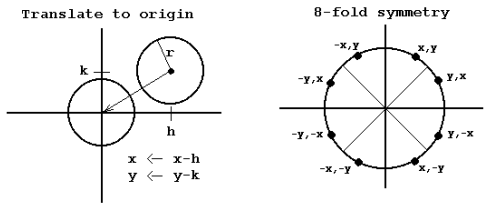
Equation:

$(x-h)^2 + (y-k)^2 = r^2$

To simplify we'll translate origin to center

Simplified Equation:

$x^2 + y^2 = r^2$



Circle has 8-fold symmetry

So only need to plot points in 1st octant

Brute Force Circle Algorithm

Suppose we have a Set8pixel() routine

```
xfin = 0.707*r
For (x=0; x<=xfin ; x++)
    {
    y = SQRT(r*r - x*x);
    Set8Pixel(round(x), round(y));
    }
```

TOO SLOW!!

The Set8Pixel(x,y) routine

```
SetPixel(x,y);
SetPixel(x,-y);
SetPixel(-x,y);
SetPixel(-x,-y);
SetPixel(y,x);
SetPixel(y,-x);
SetPixel(-y,x);
SetPixel(-y,-x);
```

Could Use Parametric Equations

```
for (theta=90; theta>=45; theta- -)
    {
    x = r*cos(theta);
    y = r*sin(theta);
    Set8Pixel(round(x), round(y));
    }
```

EVEN SLOWER!

DDA Circle Approximation

$x^2 + y^2 = r^2$

Take Derivative:

$2*x+2*y*(dy/dx) = 0$

$dy = (-x/y)*dx$

Step in x direction ($dx=1$)

$dy=-x/y$

$y = y + dy$ (approximation)

DDA Circle Algorithm

```
x=0; y=r;
xfin=0.707*r;
while (x<=xfin)
    {
    Set8Pixel(round(x), round(y));
    y = y - (x/y);
    x = x + 1;
    }
```

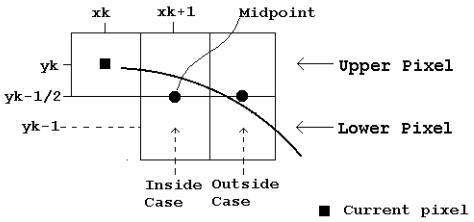
Floating Pt. Divide--STILL TOO SLOW!

Midpoint Circle Algorithm

- Extension of Bresenham ideas
- Circle equation: $x^2 + y^2 = r^2$
- Define a circle function:
  $f = x^2 + y^2 - r^2$
- f=0 ==> (x,y) is on circle
- f<0 ==> (x,y) is inside circle
- f>0 ==> (x,y) is outside circle

- We've just plotted (x_k,y_k)
- $(\Delta x > \Delta y)$, so we're stepping in x
- Next pixel is either:
  $(x_k + 1, y_k)$ -- the "top" case or
  $(x_k + 1, y_k - 1)$ -- the "bottom" case

Midpoint Circle Choices



Inside: f<0 ==> choose upper pixel
Outside: f>0 ==> choose lower pixel

- Evaluate f at midpoint
  $(x=x_k+1, y=y_k-1/2)$
- Predictor: $P_k = f(x_k+1, y_k-1/2)$
  $P_k<0$ ==> inside (choose top pixel)
  $P_k>0$ ==> outside (choose bottom pixel)
  $P_k = (x_k+1)^2 + (y_k-1/2)^2 - r^2$
- $P_k = x_k^2 + 2x_k + 5/4 + y_k^2 - y_k - r^2$
- As for Bresenham, try to get recurrence relation for P

- Top Case $(x_{k+1} = x_k + 1,\ y_{k+1} = y_k)$:

  $P_{k+1} = f(x_{k+1} + 1,\ y_{k+1} - 1/2)$
  But $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k$
  So $P_{k+1} = ((x_k+1) + 1)^2 + (y_k - 1/2)^2 - r^2$
  $P_{k+1} = x_k^2 + 4x_k + 4 + y_k^2 - y_k + 1/4 - r^2$
  But, $P_k = x_k^2 + 2x_k + 5/4 + y_k^2 - y_k - r^2$
  $\Delta P_k = P_{k+1} - P_k$
  So $\Delta Pk = 2x_k + 3$, But $x_{k+1} = x_k + 1$
  So $\Delta P_k = 2x_{k+1} + 1$

- Bottom Case $(x_{k+1} = x_k + 1,\ y_{k+1} = y_k - 1)$:

  $P_{k+1} = f(x_{k+1} + 1,\ y_{k+1} - 1/2)$
- $P_{k+1} = ((x_k+1) + 1)^2 + ((y_k-1) - 1/2)^2 - r^2$
  $= x_k^2 + 4x_k + 4 + y_k^2 - 3x_k + 9/4 - r^2$
  But $P_k = x_k^2 + 2x_k + 5/4 + y_k^2 - y_k - r^2$
  $\Delta P_k = P_{k+1} - P_k$
  So $\Delta P_k = 2x_k - 2y_k + 5$
  $\Delta P_k = 2(x_{k+1} - y_{k+1}) + 1$

- Initial P:

P0 (x0=0, y0=r)

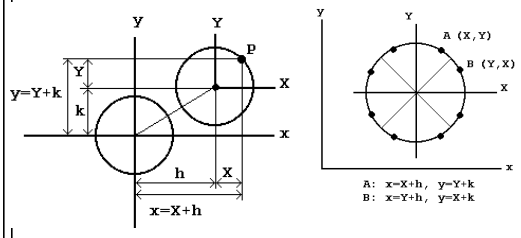
$P0 = (x0 + 1)^2 + (y0 - 1/2)^2 - r^2$

P0 = 5/4 - r --> 1-r (rounding to integer)

Midpoint Circle Algorithm

```
x=0; y=r;   P=1-r;
Set8Pixel(x,y);
while (x<y)
   {
   x = x + 1; Set8Pixel(x,y);
   if (P < 0)
      P = P + x<<1 + 1;
   else
      { y = y - 1; P = P + (x-y)<<1 + 1;}
   }
```

Circles not centered on origin



Need to redo the Set8Pixel() function

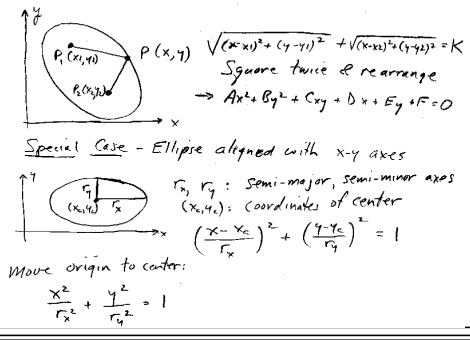
New Set8Pixel() Function

```
Set8Pixel(x,y,h,k)
{   SetPixel(x+h,y+k);
    SetPixel(x+h,-y+k);
    SetPixel(-x+h,y+k);
    SetPixel(-x+h,-y+k);
    SetPixel(y+h,x+k);
    SetPixel(y+h,-x+k);
    SetPixel(-y+h,x+k);
    SetPixel(-y+h,-x+k);
}
```

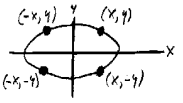
Adjusting for Aspect Ratio

- One way--adjust at pixel level
- If pixel width = w, height = h
- A.R. = h/w
- So either:
  - Multiply each x by A.R.
  - or Divide each y by A.R.

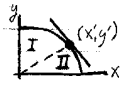
Scan Converting an Ellipse



Ellipse has 4-fold Symmetry
So use a Set4Pixel function
Only traverse 1st quadrant



Step in x until $\frac{dy}{dx} < -1$
Then Step in y



$\frac{dy}{dx}(x',y') = -1$

$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$

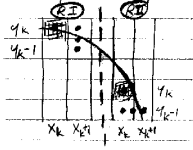
$\frac{dy}{dx} = -\frac{x r_y^2}{y r_x^2}$

$\frac{dy}{dx} = -1$ at (x',y')
So Solve for (x',y')

DDA Algorithm (Region I) -- Step in x:
$\Delta x = 1$, $\Delta y = -x r_y^2 / y r_x^2$
Each iteration: $x = x+1$
$y = y - x r_x^2 / y r_y^2$

DDA Algorithm (Region II) -- Step in y:
$\Delta y = 1$, $\Delta x = -y r_x^2 / x r_y^2$

Midpoint Ellipse Algorithm - (x_k, y_k) just plotted



Region I: $\frac{dy}{dx} > -1 \Rightarrow 2r_y^2 x < 2r_x^2 y$

Next point: $\begin{cases} (x_k+1, y_k), \text{ top} \\ (x_k+1, y_k-1), \text{ Bottom} \end{cases}$

Region II: $\frac{dy}{dx} < -1$

Next point: $\begin{cases} (x_k, y_k-1), \text{ Left} \\ (x_k+1, y_k-1), \text{ Right} \end{cases}$

Define: $P_x \equiv 2r_y^2 x$, $P_y \equiv 2r_x^2 y$
$\Rightarrow$ Region I when $P_x < P_y$

Ellipse function:
$f \equiv x^2 r_y^2 + y^2 r_x^2 - r_x^2 r_y^2$ $\begin{cases} =0 \Rightarrow (x,y) \text{ on curve} \\ <0 \Rightarrow \text{inside, choose Top} \\ >0 \Rightarrow \text{outside, choose Bot} \end{cases}$
Evaluate at $(x_k+1, y_k - \frac{1}{2})$ (midpoint)

Evaluate Ellipse function at midpoint $(x_k+1, y_k-\frac{1}{2})$:

$f_k = r_y^2 (x_k+1)^2 + r_x^2 (y_k - \frac{1}{2})^2 - r_x^2 r_y^2$

Too complex -- Try to get recurrence relation:

$f_{k+1} = f_k + \Delta f$

$x_{k+1} = x_k + 1$
$y_{k+1} = \begin{cases} y_k \text{ (top)} \\ y_k - 1 \text{ (Bottom)} \end{cases}$

So $\Delta f = f_{k+1} - f_k$

Top case: $f_{k+1} = r_y^2 [(x_k+1)+1]^2 + r_x^2 (y_k - \frac{1}{2})^2 - r_x^2 r_y^2$

Results: $\Delta f = r_y^2 (2x_k + 3)$
But $x_{k+1} = (x_k+1)$, So $\Delta f = r_y^2 (2x_{k+1} + 1)$
$\Delta f = P_{x_k} + r_y^2$

Bottom case: $f_{k+1} = r_y^2 [(x_k+1)+1]^2 + r_x^2 [(y_k - 1) - \frac{1}{2}]^2 - r_x^2 r_y^2$

Result: $\Delta f = r_y^2 (2x_k + 3) + r_x^2 (-2y_k + 2)$
But $x_{k+1} = x_k + 1$ & $y_{k+1} = y_k - 1$, So $\Delta f = r_y^2 + P_{x_k} - P_{y_k}$

Initial Values of f_{k_0}, P_{x_0}, P_{y_0}, when $x=0, y=r_y$

$f_0 = r_y^2 (0+1)^2 + r_x^2 (r_y - \frac{1}{2})^2 - r_x^2 r_y^2$
$f_0 = r_y^2 + r_x^2 (\frac{1}{4} - r_y)$

Also need initial values of P_x & P_y
$P_{x_0} = 2r_y^2 x_0 = 0$ $\quad P_{y_0} = 2r_x^2 y_0 = 2r_x^2 r_y$

Also need recurrence relations for P_x & P_y
$P_{x_k} = 2r_y^2 x_k$ $\quad P_{y_k} = 2r_x^2 y_k$
$P_{x_{k+1}} = 2r_y^2 (x_k+1)$
So $\Delta P_x = 2r_y^2$ (constant)
$P_{y_{k+1}} = \begin{cases} 2r_x^2 y_k \text{ (top)} \\ 2r_x^2 (y_k - 1) \text{ (Bottom} \end{cases}$
So $\Delta P_y = \begin{cases} 0 \text{ (Top)} \\ -2r_x^2 \text{ (Bottom)} \end{cases}$

Region II - Just plotted (x_k, y_k)
Next point $\begin{cases} (x_k, y_k-1), \text{ Left case} \\ (x_k+1, y_k-1), \text{ Right case} \end{cases}$

Midpoint: $(x_k + \frac{1}{2}, y_k - 1) \Rightarrow f = r_y^2 (x+\frac{1}{2})^2 + r_x^2 (y-1)^2 + r_x^2 r_y^2$ (predictor Alg)

Assume last point in Region I was (x', y') -

Results: $f_{init} = r_y^2 (x' + \frac{1}{2})^2 + r_x^2 (y' - 1)^2 - r_x^2 r_y^2$

Next point: $y = y - 1$
$\Delta P_y = 2r_x^2$
$f > 0 \Rightarrow \Delta f = r_x^2 - P_y$
$f < 0 \Rightarrow x = x+1$, $\Delta f = r_x^2 - P_y + P_x$

Midpoint Ellipse Alg. (Region I)

```
DPx=2*ry*ry; Dpy=2*rx*rx; x=0; y=ry; Px=0;
Py=2*rx*rx*ry; f=ry*ry+rx*rx(0.25-ry); ry2=ry*ry;
Set4Pixel(x,y);
while (px<py)  //Region I
    {
    x=x+1; Px=Px+DPx;
    if (f>0)     // Bottom case
       {y=y-1; Py=Py-Dpy; f=f+ry2+Px-Py;}
    else         // Top case
       f=f+ry2+Px;
    Set4Pixel(x,y);
    }
```

Scan Converting other 2D Curves

DDA:

y = f(x); If we can differentiate it:
dy/dx = f′(x)

Step in x for parts of curve where dy/dx < 1

x = x + 1

y = y + f′(x)

Step in y for parts of curve where dy/dx > 1

y = y + 1

x = x + 1/f′(x)

Scan Converting other 2D Curves

- Midpoint Algorithm
  - Use techniques like those used for the ellipse
  - Often we can devise a midpoint algorithm
    - Most efficient result could have several levels of recurrence