Topics

- Interactive Graphic Systems
- Drawing lines
- Drawing circles
- Filling polygons
Interactive Graphic System

Application Model

↓  ↑

Application Program

↓  ↑

Graphics System
Interactive Graphic System

Application Model
– Represents data and objects to be displayed on the output device

Application Program
– Creates, stores into, and retrieves from the application model
– Handles user-inputs
– Sends output commands to the graphics system:
  • *Which* geometric object to view (point, line, circle, polygon)
  • *How* to view it (color, line-style, thickness, texture)
Graphics System

–Intermediates between the *application program* and the *interface hardware*:
  • Output flow
  • Input flow
–Causes the application program to be *device-independent*. 
Display Hardware
CRT - Cathode Ray Tube

Cathode (electron gun)

focusing anode

deflection yoke

shadow mask
and phosphor coated screen

electron guns

shadow mask

phosphors on glass screen
Raster Scan (CRT)

Scan line
Display Hardware
FED - Field Emission Display

Anode

Phosphor

Microtips (Emitters)

Catode electrodes
Image Representation in Raster Displays

<table>
<thead>
<tr>
<th>I</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Raster Display

Graphics system

Display commands

Interaction data

Keyboard
Mouse

Frame Buffer

155  0  1
255  45  95
17  66  147
82  27  55
12  159  36

Video Controller

Display commands flow to the Display Controller, which interacts with the Frame Buffer. The Frame Buffer then sends data to the Video Controller, which in turn drives the display. Interaction data flows from the Keyboard and Mouse to the Display Controller.
Terminology

**Pixel**: Picture element.
- Smallest accessible element in picture
- Assume rectangular or circular shape

**Aspect Ratio**: Ratio between physical dimensions of a pixel (not necessarily 1)

**Dynamic Range**: The ratio between the minimal (not zero!) and the maximal light intensity a display pixel can emit

**Resolution**: The number of distinguishable rows and columns in the device. Measured in:
- Absolute values (1K x 1K) or,
- Density values (300 dpi [=dots per inch])

**Screen Space**: A discrete Cartesian coordinate system of the screen pixels

**Object Space**: The Cartesian coordinate system of the universe, in which the objects (to be displayed) are embedded
Scan Conversion

The conversion from a geometrical representation of an object to pixels in a raster display
Representations

Implicit formula:
Constraint(s) expressed as

\[ f(x, y, \ldots) = 0 \]

A k-dimensional surface embedded in n-dimensions

\[ f_i(x_1, x_2, \ldots, x_n) = 0 \quad ; \quad i = 1 \ldots n-k \]

Explicit formula:
For each x define y as \( y = f(x) \)
Good only for "functions".

Parametric formula:
Depending on free parameter(s)

\[ x = f_x(t) \]
\[ y = f_y(t) \]

For k-dimensional surface there are k free parameters
Line in 2 dimensions

• Implicit representation:
  \[ \alpha x + \beta y + \gamma = 0 \]

• Explicit representation:
  \[ y = mx + B \quad m = \frac{y_1 - y_0}{x_1 - x_0} \]

• Parametric representation:
  \[ P = \begin{pmatrix} x \\ y \end{pmatrix} \quad P = P_0 + (P_1 - P_0) t \quad t \in [0..1] \]
Scan Conversion - Lines

\[ y = mx + B \]

slope = \( m = \frac{y_1 - y_0}{x_1 - x_0} \)

offset = \( B = y_1 - mx_1 \)

Assume \(|m| \leq 1\)
Assume \(x_0 \leq x_1\)

Basic Algorithm
For \( x = x_0 \) to \( x_1 \)
\[ y = mx + B \]
PlotPixel(x,round(y))
end;

For each iteration: 1 float multiplication, 1 addition, 1 round
Scan Conversion - Lines

Incremental Algorithm

\[ y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B = y_i + m\Delta x \]

if \( \Delta x = 1 \) then

\[ y_{i+1} = y_i + m \]

Algorithm

\[ y = y_0 \]

For \( x = x_0 \) to \( x_1 \)

\[ \text{PlotPixel}(x, \text{round}(y)) \]

\[ y = y + m \]

end;
Scan Conversion - Lines

\[(x_i+1, \text{Round}(y_i+m))\]

\[(x_i, y_i)\]

\[(x_i, \text{Round}(y_i))\]

\[(x_{i+1}, y_i+m)\]
Pseudo Code for Basic Line Drawing

Assume $x_1 > x_0$ and line slope absolute value is $\leq 1$

```
Line(x_0, y_0, x_1, y_1)
begin
  float dx, dy, x, y, slope;
  dx := x_1 - x_0;
  dy := y_1 - y_0;
  slope := dy/dx;
  y := y_0;
  for x := x_0 to x_1 do
    begin
      PlotPixel(x, Round(y));
      y := y + slope;
    end;
end;
```
Basic Line Drawing

**Symmetric Cases:**

\[ |m| \geq 1 \]

\[
x = x_0
\]

For \( y = y_0 \) to \( y_1 \)

\[
\text{PlotPixel}(\text{round}(x), y)
\]

\[
x = x + \frac{1}{m}
\]

end;

**Special Cases:**

\[ m = \pm 1 \text{ (diagonals)} \]

\[ m = 0, \infty \text{ (horizontal, vertical)} \]

**Symmetric Cases:**

if \( x_0 > x_1 \) for \(|m| \leq 1\) or \( y_0 > y_1 \) for \(|m| \geq 1\)

\[
\text{swap}((x_0, y_0), (x_1, y_1))
\]

**For each iteration:**

- 1 addition, 1 rounding

**Drawbacks:**

- Accumulated error
- Floating point arithmetic
- Round operations
Midpoint (Bresenham) Line Drawing

Assumptions:
• \( x_0 < x_1 \), \( y_0 < y_1 \)
• \( 0 < \text{slope} < 1 \)

Given \((x_p,y_p)\), the next pixel is \( E = (x_p+1,y_p) \) or \( \text{NE} = (x_p+1,y_p+1) \)

**Bresenham:** \( \text{sign}(M-Q) \) determines NE or E

\[
M = (x_p +1,y_p +1/2)
\]
Midpoint (Bresenham) Line Drawing

\[
y = \frac{dy}{dx} \ x + B
\]

Implicit form of a line:

\[
f(x,y) = ax + by + c = 0
\]

\[
f(x,y) = dy x - dx y + B \ dx = 0
\]

Decision Variable:

\[
d = f(M) = f(x_p +1,y_p +1/2) = a(x_p +1) + b(y_p +1/2) + c
\]

- choose NE if \( d > 0 \)
- choose E if \( d \leq 0 \)

(a>0)
Midpoint (Bresenham) Line Drawing

What happens at $x_p + 2$?

If $E$ was chosen at $x_p + 1$

$$M = (x_p + 2, y_p + 1/2)$$

$$d_{\text{new}} = f(x_p + 2, y_p + 1/2) = a(x_p + 2) + b(y_p + 1/2) + c$$

$$d_{\text{old}} = f(x_p + 1, y_p + 1/2) = a(x_p + 1) + b(y_p + 1/2) + c$$

$$d_{\text{new}} = d_{\text{old}} + a = d_{\text{old}} + dy$$

If $NE$ was chosen at $x_p + 1$

$$M = (x_p + 2, y_p + 3/2)$$

$$d_{\text{new}} = f(x_p + 2, y_p + 3/2) = a(x_p + 2) + b(y_p + 3/2) + c$$

$$d_{\text{old}} = f(x_p + 1, y_p + 1/2) = a(x_p + 1) + b(y_p + 1/2) + c$$

$$d_{\text{new}} = d_{\text{old}} + a + b = d_{\text{old}} + dy - dx$$

$$d_{\text{new}} = d_{\text{old}} + \Delta_{NE}$$
Midpoint (Bresenham) Line Drawing

**Initialization:**

First point = \((x_0, y_0)\), first MidPoint = \((x_0+1, y_0+1/2)\)

\[
d_{\text{start}} = f(x_0+1, y_0+1/2) = a(x_0 +1) + b(y_0 +1/2) + c
\]

\[
= ax_0 + by_0 + c + a + b/2
\]

\[
= f(x_0,y_0) + a + b/2 = a + b/2
\]

\[
d_{\text{start}} = dy - dx/2
\]

**Enhancement:**

To eliminate fractions, define:

\[
f(x,y) = 2(ax + by + c) = 0
\]

\[
d_{\text{start}} = 2dy - dx
\]

\[
\Delta_E = 2dy
\]

\[
\Delta_{NE} = 2(dy-dx)
\]
Midpoint (Bresenham) Line Drawing

• The sign of \( f(x_{0}+1,y_{0}+1/2) \) indicates whether to move East or North-East.

• At the beginning \( d=f(x_{0}+1,y_{0}+1/2)=2dy-dx \).

• The increment in \( d \) (after this step) is:
  
  - If we moved East: \( \Delta_E=2dy \)
  - If we moved North-East: \( \Delta_{NE}=2dy-2dx \)

Comments:

• Integer arithmetic (\( dx \) and \( dy \) are integers)
• One addition for each iteration
• No accumulated errors
• By symmetry, we deal with \( 0>\text{slope}>-1 \)
Pseudo Code for Midpoint Line Drawing

\[ \text{Line}(x_0, y_0, x_1, y_1) \]

\begin{verbatim}
begin
  int dx, dy, x, y, d, \( \Delta_E \), \( \Delta_{NE} \);
  \( x := x_0 \); \( y := y_0 \);
  \( dx := x_1 - x_0 \); \( dy := y_1 - y_0 \);
  \( d := 2*dy-dx \);
  \( \Delta_E := 2*dy \); \( \Delta_{NE} := 2*(dy-dx) \);
  PlotPixel(x, y);
  while (x < x_1) do
    if (d < 0) then
      d := d + \( \Delta_E \);
      x := x + 1;
    end;
    else
      d := d + \( \Delta_{NE} \);
      x := x + 1;
      y := y + 1;
    end;
    PlotPixel(x, y);
  end;
end;
\end{verbatim}

Assume \( x_1 > x_0 \) and \( 0 < \text{slope} \leq 1 \)
Scan Conversion - Circles

Implicit representation (centered at the origin, radius $R$):

$$x^2 + y^2 - R^2 = 0$$

Explicit representation:

$$y = \pm \sqrt{R^2 - x^2}$$

Parametric representation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} R \cos(t) \\ R \sin(t) \end{pmatrix} \quad t \in [0 \ldots 2\pi]$$
Scan Conversion - Circles

Basic Algorithm

For $x = -R \text{ to } R$
\[ y = \sqrt{R^2-x^2} \]
PlotPixel($x$,\text{round}(y))
PlotPixel($x$,-\text{round}(y))
end;

Comments:
- Square-root operations are expensive
- Floating point arithmetic
- Large gap for $x$ values close to $R$
Scan Conversion - Circles

Exploiting Eight-Way Symmetry

For a circle centered at the origin:
If \((x,y)\) is on the circle then

\((y,x)\) \((y,-x)\) \((x,-y)\) \((-x,-y)\) \((-y,-x)\) \((-y,x)\) \((-x,y)\)

are on the circle as well.

Therefore we need to compute only one octant \(45^\circ\) segment.
Scan Conversion - Circles

CirclePoints(x, y)
begin
  PlotPixel(x,y);
  PlotPixel(y,x);
  PlotPixel(y,-x);
  PlotPixel(x,-y);
  PlotPixel(-x,-y);
  PlotPixel(-y,-x);
  PlotPixel(-y,x);
  PlotPixel(-x,y);
end;
Circle Midpoint (for one octant)

(The circle is located at (0,0) with radius R)

- We start from \((x_0,y_0)=(0,R)\)
- One can move either **East** or **South-East**
- Again, \(d(x,y)\) will be a threshold criteria at the midpoint
Circle Midpoint (for one octant)

Threshold Criteria:

\[ d(x,y) = f(x,y) = x^2 + y^2 - R^2 = 0 \]

- \( f(x,y) < 0 \)
- \( f(x,y) > 0 \)
Circle Midpoint (for one octant)

• At the beginning
  \[ d_{\text{start}} = d(x_0+1,y_0-1/2) \]
  \[ = d(1,R-1/2) = 5/4 - R \]

• If \( d < 0 \) we move \textbf{East}:
  \[ \Delta_E = d(x_0+2,y_0-1/2) - d(x_0+1,y_0-1/2) = 2x_0+3 \]

• If \( d > 0 \) we move \textbf{South-East}:
  \[ \Delta_{SE} = d(x_0+2,y_0-3/2) - d(x_0+1,y_0-1/2) = 2(x_0-y_0)+5 \]

• \( \Delta_E \) and \( \Delta_{SE} \) are not constant anymore

• Since \( d \) is incremented by integer values, we can use
  \[ d_{\text{start}} = 1-R \]
  yielding an integer algorithm. This has no affect on the threshold criteria.
Pseudo Code for Circle Midpoint

MidpointCircle (R)
begin
int x, y, d;
x := 0;
y := R;
d := 5.0/4.0-R;
CirclePoints(x,y);
while ( y > x ) do
    if ( d < 0 ) then /* East */
        d := d+2x+3;
        x := x+1;
    end;
    else /* South East */
        d := d+2(x-y)+5;
        x := x+1;
        y := y-1;
    end;
    CirclePoints( x,y ); /* Mirror to create the other seven octants */
end;
Pseudo Code for Circle Midpoint

MidpointCircle (R)
begin
    int x, y, d;
    x := 0;
    y := R;
    d := 1-R;                        /* originally  d := 5.0/4.0 - R  */
    CirclePoints(x,y);
    while ( y>x ) do
        if ( d<0 ) then            /* East */
            d := d+2x+3;        /* Multiplication! */
            x := x+1;
        end;
        else                            /* South East */
            d := d+2(x-y)+5;    /* Multiplication! */
            x := x+1;
            y := y-1;
        end;
    CirclePoints( x,y );      /* Mirror to create the other seven octants */
end;
Pseudo Code for Circle Midpoint

MidpointCircle (R)
begin
  int x, y, d;
  x := 0;
y := R;
d := 1-R;               /* originally  d := 5.0/4.0 - R  */
  $\Delta_E = 3$;
  $\Delta_{SE} = -2R+5$;
  CirclePoints(x,y);
  while ( y>x ) do
    if ( d<0 ) then  /* East */
      d := d+$\Delta_E$;
      $\Delta_E := \Delta_E+2$;
      $\Delta_{SE} := \Delta_{SE}+2$;
      x := x+1;
    end;
    else  /* South East */
      d := d+$\Delta_{SE}$;
      $\Delta_E := \Delta_E+2$;
      $\Delta_{SE} := \Delta_{SE}+4$;
      x := x+1;
y := y-1;
    end;
  CirclePoints( x,y );  /* Mirror to create the other seven octants */
end;
Circle Midpoint
Polygon Fill

Representation:

Polygon = $V_0, V_1, V_2, \ldots, V_n$

$V_i=(x_i,y_i)$ - vertex

$E_i=(v_i,v_{i+1})$ - edge

$E_n=(v_n,v_0)$
Scan Conversion - Polygon Fill

**Problem:** Given a closed 2D polygon, fill its interior with a specified color, on a graphics display

**Assumption:** Polygon is simple, i.e. no self intersections, and simply connected, i.e. without holes

**Solutions:**
- Flood fill
- Scan Conversion
Flood Fill Algorithm

• Let $P$ be a polygon with $n$ vertices, $v_0 \ldots v_{n-1}$
• Denote $v_n = v_0$
• Let $c$ be a color to paint $P$
• Let $p = (x, y)$ be a point in $P$
Flood Fill Algorithm

\[
\text{FloodFill}(P, x, y, c) \\
\quad \text{if } (\text{OnBoundary}(x, y, P) \text{ or } \text{Colored}(x, y, c)) \\
\quad \text{then return;} \\
\quad \text{else begin} \\
\quad \quad \text{PlotPixel}(x, y, c); \\
\quad \quad \text{FloodFill}(P, x+1, y, c); \\
\quad \quad \text{FloodFill}(P, x, y+1, c); \\
\quad \quad \text{FloodFill}(P, x, y-1, c); \\
\quad \quad \text{FloodFill}(P, x-1, y, c); \\
\quad \text{end;}
\]

Slow algorithm due to recursion, needs initial point
Question: How do we know if a given point is inside or outside a polygon?
Scan Conversion – Basic Algorithm

- Let P be a polygon with n vertices, $v_0 \ldots v_{n-1}$
- Denote $v_n = v_0$
- Let c be a color to paint P

ScanConvert ($P, c$)

For $j := 0$ to ScreenYMax do
  $I :=$ points of intersection of edges from $P$ with line $y=j$;
  Sort $I$ in increasing $x$ order and fill with color $c$ alternating segments;
end;

Question: How do we find the intersecting edges?
Scan Conversion – Fill Polygon

What happens in these cases?
Scan Conversion – Fill Polygon

Intersections at pixel coordinates

**Rule:** In the odd/even count, we count $y_{\text{min}}$ vertices of an edge, but not $y_{\text{max}}$ vertices

Vertex B is counted once because $y_{\text{min}}$ of (A,B)
Vertex E is not counted because $y_{\text{max}}$ of both (D, E) and (E, F)
Vertex H is counted twice because $y_{\text{min}}$ of both (G, H) and (H, I)
Fill Polygon – Optimized Algorithm

Uses a list of “active” edges A (edges currently intersecting the scan line)

\[
\text{ScanConvert}(P,c)
\]

Sort all edges \(E=\{E_j\}\) in increasing \(\text{MinY}(E_j)\) order.

\(A := \emptyset;\)

For \(k := 0\) to \(\text{ScreenYMax}\) do

For each \(E_j \in E,\)

if \(\text{MinY}(E_j) \leq k\) \(A := A \cup E_j; \ E = E - E_j\)

For each \(E_j \in A,\)

if \(\text{MaxY}(E_j) \leq k\) \(A := A - E_j\)

\(I:\) Points of intersection of members from \(A\) with line \(y = k;\)

Sort \(I\) in increasing \(x\) order and draw with color \(c\) alternating segments;

end;
Fill Polygon – Optimized Algorithm

Implementation with linked lists

\[ E \quad y_{\text{min}} \quad 5 \quad 7 \]

\[ A \quad 3 \quad 12 \quad 4 \quad 10 \quad 15 \]

\[ y_{\text{max}} \]
### Flood Fill vs. Scan Conversion

<table>
<thead>
<tr>
<th>Flood Fill</th>
<th>Scan Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Very simple</td>
<td>• More complex</td>
</tr>
<tr>
<td>• Requires a seed point</td>
<td>• No seed point is required</td>
</tr>
<tr>
<td>• Requires large stack size</td>
<td>• Requires small stack size</td>
</tr>
<tr>
<td>• Common in paint packages</td>
<td>• Used in image rendering</td>
</tr>
</tbody>
</table>