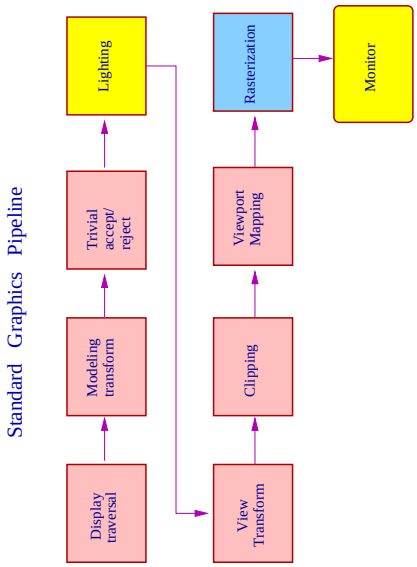
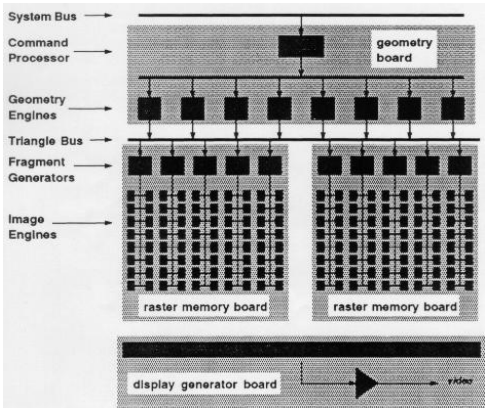
Viewing Pipeline



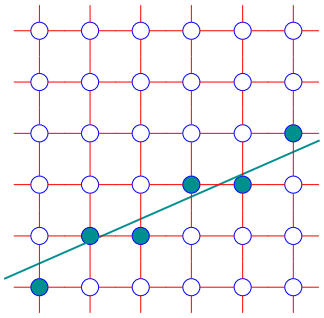
Slide 1

Graphics Hardware



Slide 2

Scan Line Conversion



- Compute the coordinates of pixels that need to be illuminated to draw e.
- Pixels should be as close to e as possible.

Assume $0 \leq m \leq 1$

- One pixel is drawn in each column.

How does one handle other values of m?

Slide 3

A Naive Algorithm

- e: A segment.
- $p = (x_1, y_1)$: Left endpoint of e.
- $q = (x_2, y_2)$: Right endpoint of e.

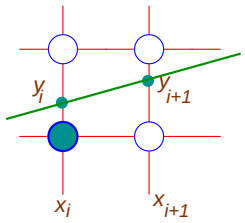
$$y = mx + B.$$
$$m = \frac{y_2 - y_1}{x_2 - x_1}, \quad B = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}.$$

```
for(x = x1; x ≤ x2; x++)
  {
    y = mx + B;
    writePixel (x, round (y));
  }
```

round (y) = $\lfloor y + 0.5 \rfloor$

Slide 4

DDA Algorithm



☆ Drawing the segment from left to right.

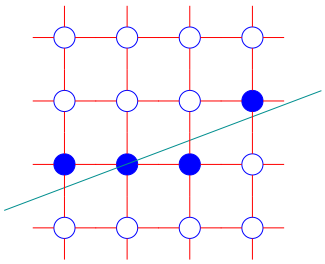
☆ Incrementing x at each step.

(x_i, y_i): Point computed in the i-th step.

$$\begin{aligned} x_{i+1} &= x_i + 1, \\ y_{i+1} &= m(x_i + 1) + B \\ &= mx_i + B + m \\ &= y_i + m. \end{aligned}$$

Slide 5

DDA Algorithm



$y = y_1$
`for`$(x = x_1;\ x \le x_2;\ x++)$
{
`writePixel` $(x,$ `round` $(y));$
$y = y + m;$
}

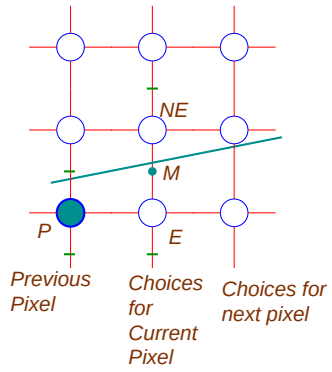
☆ Computing $\lfloor \quad \rfloor$ is expensive.

☆ Involves floating-point arithmetic.

☆ At each step we need to decide which of the two pixels should be selected.

Slide 6

Midpoint Algorithm



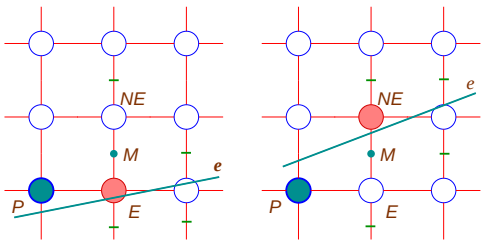
P: Previous pixel drawn.

(x_P, y_P): Coordinates of P.

$$M = (x_P + 1, y_P + 0.5)$$

Slide 7

Midpoint Algorithm



M *above* $e \Longrightarrow$ Choose $E = (x_P + 1, y_P)$.

M *below* $e \Longrightarrow$ Choose $NE = (x_P + 1, y_P + 1)$.

Slide 8

Midpoint Algorithm

$$
\begin{aligned}
y &= mx + B = \frac{\Delta y}{\Delta x}x + B \\
&\quad \Delta x = x_2 - x_1 \quad \Delta y = y_2 - y_1 \\
\Delta x \cdot y &= \Delta y \cdot x + \Delta x \cdot B \\
&\quad ax + by + c = 0 \\
&\quad a = \Delta y,\ b = -\Delta x, c = \Delta x \cdot B
\end{aligned}
$$

Let $F(x, y) = ax + by + c$.

$F(x, y) > 0$ (x, y) above e

$F(x, y) < 0$ (x, y) below e

$F(x, y) = 0$ (x, y) on e

Does $M = (x_P + 1, y_P + 0.5)$ lie above e?

$$
\begin{aligned}
F(M) &= F(x_P + 1, y_P + 0.5) \\
&= \underbrace{a(x_P + 1) + b(y_P + 0.5) + c}_{D}
\end{aligned}
$$

- ☆ $D \geq 0$: E is chosen.
- ☆ $D < 0$: NE is chosen.

Slide 9

Midpoint Algorithm

Suppose we are currently drawing the pixel in the column x_{i+1}.

M_{curr}: $(x_P + 1, y_P + 0.5)$.

D_{curr}: $a(x_P + 1) + b(y_P + 0.5) + c$.

If $E = (x_P + 1, y_P)$ is chosen:

- ☆ either $(x_P + 2, y_P)$ or $(x_P + 2, y_P + 1)$ chosen in the next step.
- ☆ $M_{new} = (x_P + 2, y_P + 0.5)$.

$$
\begin{aligned}
D_{new} &= F(x_P + 2, y_P + 0.5) \\
&= a(x_P + 2) + b(y_P + 0.5) + c \\
&= a(x_P + 1) + b(y_P + 0.5) + c + a \\
&= D_{curr} + a.
\end{aligned}
$$

Slide 10

Midpoint Algorithm

If $NE = (x_P + 1, y_P + 1)$ is chosen :

- either $(x_P + 2, y_P + 1)$ or $(x_P + 2, y_P + 2)$ chosen in the next step.
- $M_{new} = (x_P + 2, y_P + 1.5)$.

$$
\begin{aligned}
D_{new} &= F(x_P + 2, y_P + 1.5) \\
&= a(x_P + 2) + b(y_P + 1.5) + c \\
&= a(x_P + 1) + b(y_P + 0.5) + c + a + b \\
&= D_{curr} + a + b.
\end{aligned}
$$

$$
\begin{aligned}
F(x_1 + 1, y_1 + 0.5) &= a(x_1 + 1) + b(y_1 + 0.5) + c \\
&= ax_1 + by_1 + c + a + b/2 \\
&= a + b/2.
\end{aligned}
$$

Slide 11

Midpoint Algorithm

$y = y_1$; $\quad D = a + 0.5b$

```
for(x = x1 + 1; x ≤ x2; x++)
    {
        if D ≥ 0
        D = D + a;
        else {
                y = y + 1;
                D = D + a + b;
                }
        writePixel (x, y);
        }
```

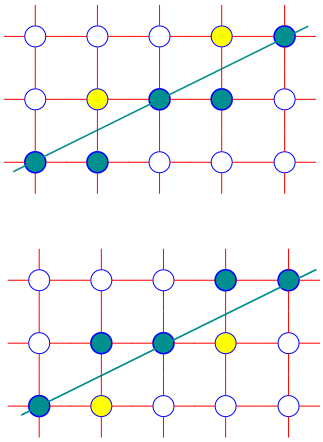
- ☆ Initial value of D is not an integer.
- ☆ $2F(x, y) \geq 0 \Leftrightarrow F(x, y) \geq 0$.
- ☆ $2F(x_1 + 1, y_1 + 0.5) = 2a + b$ is an integer Compute $2F(M)$ at each step.

Slide 12

Additional Issues

☆ The line drawn should be independent of whether we draw it from p to q or from q to p.

- Unique choices except when $D = 0$.
- Choose SW instead of W when moving from right to left.

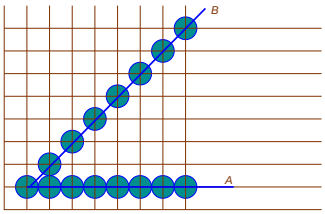


Slide 13

Additional Issues

☆ Consider lines A and B

- $|B| = \sqrt{2}|A|$
- Same number of pixels are drawn, so intensity of B is less than that of A.
- Vary intensity with slope.

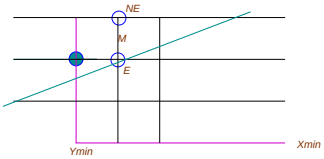


Slide 14

Additional Issues

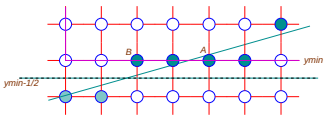
☆ Clipping at a vertical line

- Choose the initial value of D carefully.



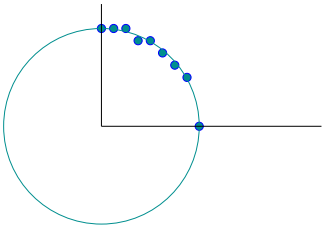
☆ Clipping at a horizontal line:

- If we clip before scan conversion, A is the first pixel drawn.
- If clip after scan conversion, B is the first pixel drawn.



Slide 15

Circle Drawing

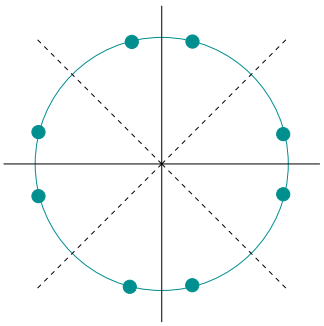


☆ Draw the circle for orientation in the range $[\pi/4, \pi/2]$.

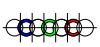
☆ Use symmetry to draw the other three portions of the circle.

Slide 16

Circle Drawing

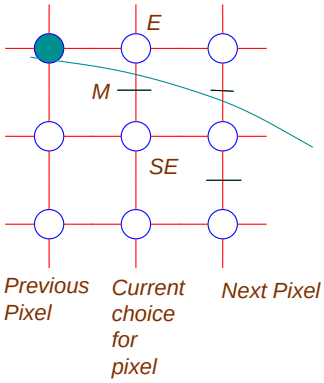


- ☆ Use the midpoint algorithm.
- ☆ If the midpoint lies inside the circle, draw E; otherwise draw SE.



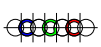
Slide 17

Circle Drawing



$$F(x, y) = x^2 + y^2 - R^2$$

- ☆ $F(M) \leq 0 \Rightarrow E$ is chosen.
- ☆ $F(M) > 0 \Rightarrow SE$ is chosen.
- ☆ $D_{curr} = (x_P + 1)^2 + (y_P - 0.5)^2 - R^2$.



Slide 18

Circle Drawing

If E is chosen

$$\begin{aligned} D_{new} &= F(x_P + 2, y_P - 0.5) \\ &= (x_P + 1)^2 + 2(x_p + 1) + 1 + \\ &\quad (y_P - 0.5)^2 - R^2 \\ &= D_{curr} + 2x_P + 3. \end{aligned}$$

If SE is chosen

$$\begin{aligned} D_{new} &= F(x_P + 2, y_P - 1.5) \\ &= (x_P + 1)^2 + 2(x_p + 1) + 1 + \\ &\quad (y_P - 0.5)^2 - 2(y_P - 0.5) + 1 + R^2 \\ &= D_{curr} + 2x_P - 2y_P + 5. \end{aligned}$$

$$\begin{aligned} D_{init} &= F(1, R - 1.5) \\ &= 1 + (R^2 - R + 0.25) = 1.25 - R. \end{aligned}$$

Slide 19

Circle Drawing

```
x = x1;
D = 1.25 − R;
while x < y
      {
        if D ≤ 0
        {
        D = D + 2x + 3;
        x = x + 1;
        }
        else {
                D = D + 2x − 2y + 5;
                y = y − 1; x = x + 1;
                }
        writePixel (x, y);
        }
```


Slide 20

Second Order Derivatives

We compute two functions

$$\begin{aligned} \Delta E(x,y) &= 2x+3 \\ \Delta SE(x,y) &= 2(x-y)+5 \end{aligned}$$

Maintain $\Delta E, \Delta SE$ also recursively;

If E is chosen, $M_{new} = (x_P+1, y_P)$.

$$\begin{aligned} \Delta E(x_P+1,y_P) &= 2(x_P+1)+3 \\ &= \Delta E(x_P,y_P)+2 \\ \Delta SE(x_P+1,y_P) &= 2(x_P+1-y_P)+5 \\ &= \Delta SE(x_P,y_P)+2. \end{aligned}$$

If SE is chosen, $M_{new} = (x_P+1, y_P-1)$

$$\begin{aligned} \Delta E(x_P+1,y_P-1) &= 2(x_P+1)+3 \\ &= \Delta E(x_P,y_P)+2. \\ \Delta SE(x_P+1,y_P) &= 2(x_P+1-y_P+1)+5 \\ &= \Delta SE(x_P,y_P)+4. \end{aligned}$$

Slide 21

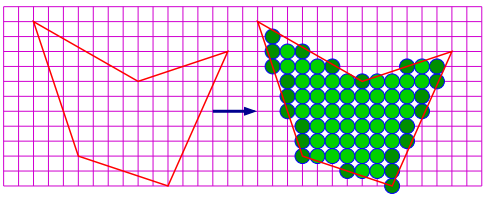
Second Order Derivatives

```
while x < y
      {
        if D ≤ 0
        {
        D = D + ΔE;
        ΔE = ΔE + 2;
        ΔSE = ΔSE + 2;
        else {
              D = D + ΔSE;
              ΔE = ΔE + 2;
              ΔSE = ΔSE + 4;
              y = y − 1;
              }
        x = x + 1;
        writePixel (x, y);
        }
```

Slide 22

Polygon Scan Conversion

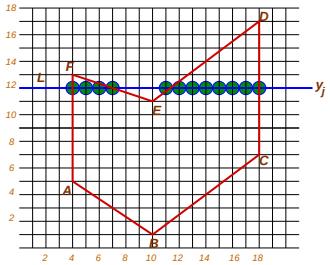
Convert edge lists to 2D field of pixels



- ☆ Draw pixels that belong to the polygon.
- ☆ Must draw pixels that lie
  - on the vertices
  - on the edges
  - in the interior of polygons.
- ☆ Must handle degenerate cases.

Slide 23

Polygon Scan Conversion



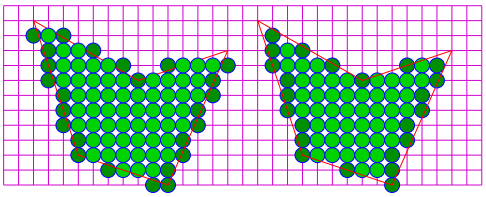
- ☆ Sweep the plane from bottom to top.
- ☆ For each scan line $L : y = y_j$
  - Compute the intersection of P with L,
  - Draw the corresponding pixels.

$P \cap L$: Set of horizontal line segments
$(p_1, q_1), \ldots (p_k, q_k)$.

```
for(i = 1; i ≤ k; i++)
   for(x = round(p_i); x ≤ round(q_i); x++)
      writePixel (x, y_j);
```

Slide 24

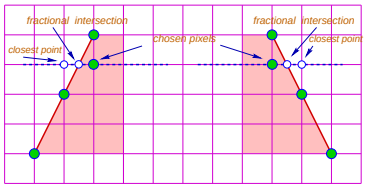
Polygon Scan Conversion



- ☆ Algorithm picks pixels closest to edges.
- ☆ Some pixels may lie outside P.
- ☆ Will interfere with polygons adjacent to P.
- ☆ Round p_i up to the next integer and q_i down to the previous integer.

Slide 25

Polygon Scan Conversion

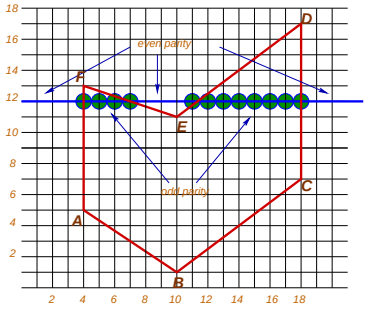



```
for (i = 1; i ≤ k; i++)
for(x = ⌈p_i⌉; x ≤ ⌊q_i⌋; x++)
    writePixel (x, y_j);
```

Slide 26

Algorithmic Details

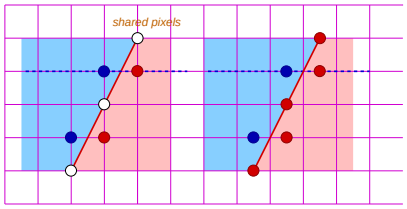
- ☆ Computing and rounding off intersection points of the scan line and edges of P.
- ☆ Sorting the intersection points left to right.
- ☆ Computing pixels lying inside $P \cap L$
  - *Parity* of a pixel is the number of polygon edges crossing the scan line to the left of the given pixel.
  - Points with odd parity lie inside P if L does not pass thru a vertex.



Slide 27

Algorithmic Details

- ☆ Pixels lying on an edge belong to the interior of the polygon.
- ☆ What if an edge is shared by two polygons?
  - Pixel lies in the interior of two polygons.



σ: Pixel lying on an edge

- ☆ Draw σ, if σ is the left endpoint of a span
- ☆ Do not draw σ, if σ is the right endpoint of a span

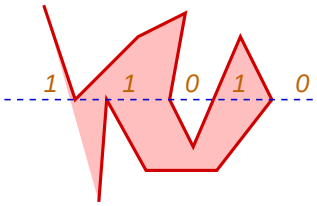
Lone polygons have pixels missing on right edges!

Slide 28

Algorithmic Details

Scan-line through a vertex: Use the following rules for counting the parity of pixels:

- ☆ Do not include horizontal edges.
- ☆ Count the non-horizontal edges whose endpoints do not lie on the scan line.
- ☆ Count the edges whose lower endpoints lie on the scan line.
- ☆ Do not count the edges whose upper endpoints lie on the scan line.

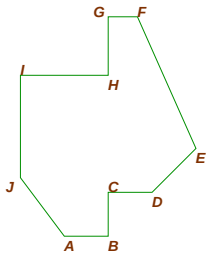


Slide 29

Algorithmic Details

Horizontal edges: Follow the above rule:

- ☆ Do not include horizontal edges in counting parity.
- ☆ Bottom horizontal edges are drawn.
- ☆ Top horizontal edges are not drawn.



Slide 30

Computing Intersections

e: An edge with endpoints (x_1, y_1), (x_2, y_2)

$$e : y = mx + B \qquad m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$y_{i+1} = y_i + 1, \quad x_{i+1} = x_i + \frac{1}{m}$$

Assume $m \geq 1$.

Example: e: $(8,8), (12,18)$

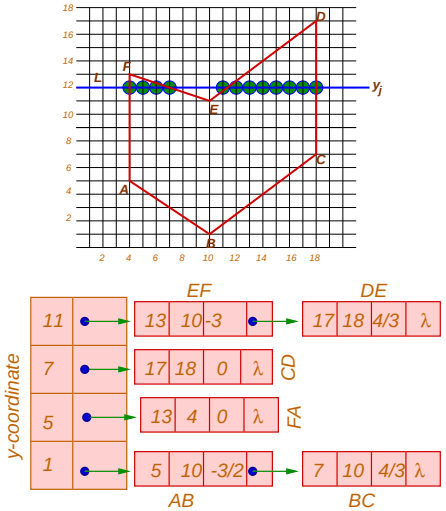
$$(8,8),\ \left(8\frac{2}{5}, 9\right)\ \left(8\frac{4}{5}, 10\right)\ \left(9\frac{1}{5}, 11\right)$$

$\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$
$I = 0$; $x = x_1$
`for`$(y = y_1;\ y \leq y_2;\ y++)$
  `writePixel` (x, y);
  $I = I + \Delta x$;
  `if` $I \geq \Delta y$
    {
      $x = x + 1$; $I = I - \Delta y$;
    }

- ☆ Store the list of polygon edges (*Edge Table*).
- ☆ Maintain the list of polygon edges intersecting the sweep line (*Active Edge Table*).

Slide 31

Edge Table



Slide 32

Edge Table

- ☆ Store all polygon edges, sorted by the y-coordinates of their lower endpoints:
  - For each y_i, there is a list that stores all edges whose lower endpoints lie on the line $y = y_i$.
- ☆ All edges with the same y-coordinate of their lower endpoints are sorted from left to right.
- ☆ For each edge e, store
  - y-coordinate of the upper endpoint,
  - x-coordinate of the lower endpoint,
  - $1/m_e$.
- ☆ Horizontal edges are not stored.

Slide 33

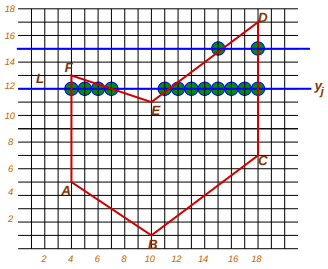
Active Edge Table

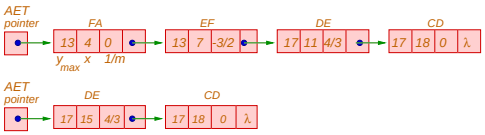
During the sweep maintain the following information.

- ☆ Maintain the list of polygon edges intersected by the scan line.
- ☆ Edges are sorted from left to right.
- ☆ Store the following information for each edge e:
  - y-coordinate of the upper endpoint of e.
  - x-coordinate of the pixel to be drawn for e at the current scan line.
  - $1/m_e$ (m_e: slope of e).
  - Can also store $\Delta x, \Delta y, I$.

Slide 34

Active Edge Table





Slide 35

Sweep Algorithm

AET = $\emptyset$
`for` $(y = y_{\min}; y \leq y_{\max}; y++)$

{

Delete the edges from AET whose upper endpoints are at y.

Update the x intercepts for edges in AET

Fill in desired pixels on scan line y

Add edges from ET to AET whose lower endpoints lie at y

}

Slide 36