Rasterization

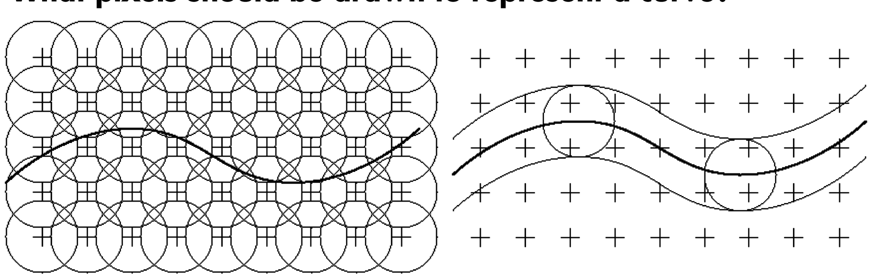
Lines and circles

- Scan conversion = sampling
- Representations of curves and lines
- Line drawing algorithms
  - DDA
  - Bresenham's algorithm
- Antialiasing
- Circle drawing algorithms

Polygons (next lecture)

Scan Conversion as Sampling

What pixels should be drawn to represent a curve?

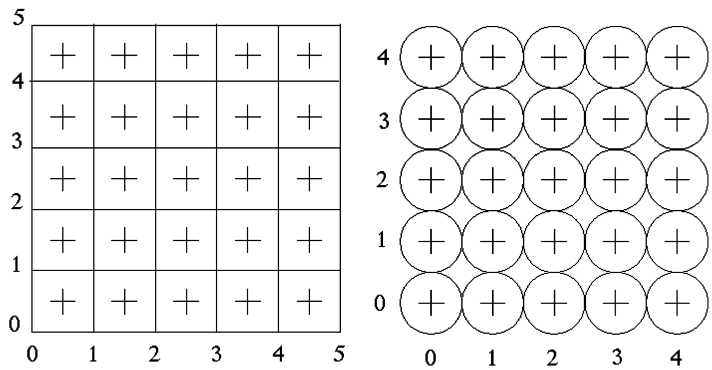


Overlap **Sweep**

Shape convolution

$$(\text{impulse} \otimes \text{pixel}) \times \text{curve} = (\text{curve} \otimes \text{pixel}) \times \text{impulse}$$

Pixel Conventions



Graph Paper
Half-Integer Pixel Centers

Pixel Spot Function
Integer Pixel Centers

A Pixel is *Not* a Little Square, A Pixel is *Not* a Little Square, ...

Mathematical Curves

	Parametric Equation	Implicit Equation
General	$(x(t), y(t))$	$f(x,y)\begin{cases} <0 & \text{inside} \\ =0 & \text{on} \\ >0 & \text{outside} \end{cases}$
Line	$x(t) = x_1 + t(x_2 - x_1)$ $y(t) = y_1 + t(y_2 - y_1)$	$f(x,y) = ax + by + c$ $a = y_2 - y_1$ $b = x_1 - x_2$ $c = x_2 y_1 - x_1 y_2$
Circle	$x(t) = x_c + r\cos 2\pi t$ $y(t) = y_c + r\sin 2\pi t$	$f(x,y) = (x - x_c)^2 + (y - y_c)^2 - r^2$

DDA

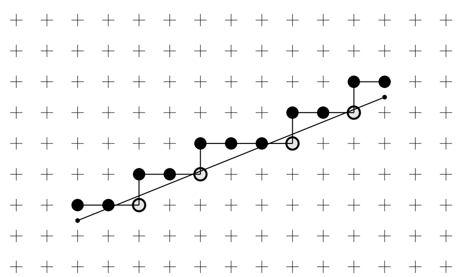
```
line(int x1, int y1, int x2, int y2)
{
  int dx = x2-x1, dy = y2-y1;
  int n = max(abs(dx),abs(dy));
  float dt = n, dxdt = dx/dt, dydt = dy/dt;

  int x = x1;
  int y = y1;
  while( n-- ) {
    point(round(x),round(y));
    x += dxdt;
    y += dydt;
  }
}
```

DDA = Digital Differential Analyzer

= finite differences

Tracing a Line



```
while( n-- ) {
  fragment(x,y);
  move right;
  if( below line )
    move up;
```

Bresenham's Algorithm

```
vect(int x, int y, int n,
     value a, value b, value c)
{
  value e;
  c = 2*c+b; a = 2*a; b = 2*b;
  e = ax+by+c;

  while( n-- ) {         // 0<=e<b
    fragment(x,y);       // |a|<|b|

    x += 1; e += a;      // a < 0
    if( e < 0 ) {
      y += 1; e += b;    // b > 0
    }
}
```

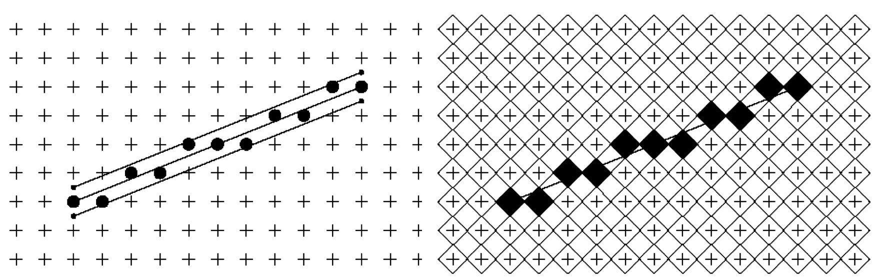
- **Line equation**
- **Finite differences**

  f(x+1,y)-f(x,y) = a
  f(x,y+1)-f(x,y) = b

- **Translate line down 1/2 pixel (or add 1/2 to y)**

  y'= y + 1/2
  c' = c + 1/2 b

Bresenham Pixels

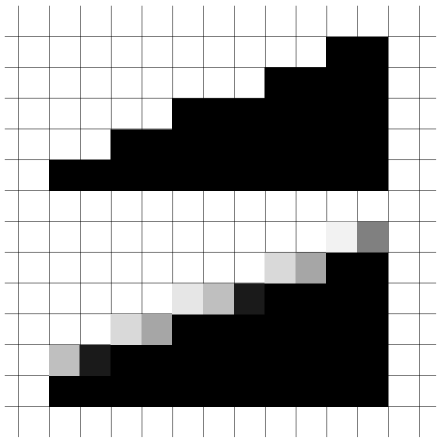


$$d_{\uparrow} = \frac{e}{2b}$$

$$0 \le e < 2b$$
$$0 \le d_{\uparrow} < 1$$

$$d_{\perp} = d_{\uparrow} \cos\theta = \frac{e}{2b} \frac{b}{\sqrt{a^2 + b^2}}$$

Jaggy vs. Smooth Lines

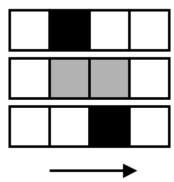


Trade-off

- Intensity resolution
- Spatial resolution

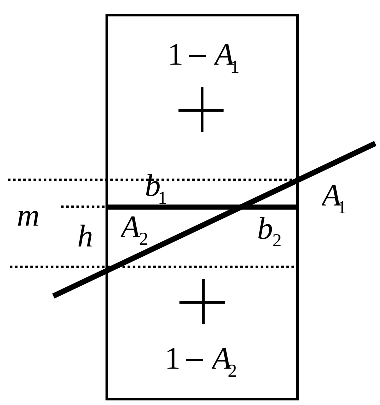
Prefiltering

- Area average
- Coverage



Imagine a
1 pixel vert line
moving from left to right

Square Pixel Filter



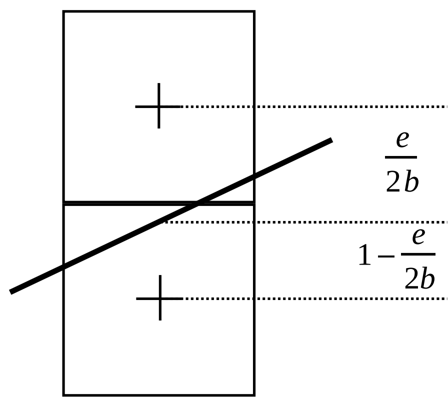
$$\frac{b_1}{h} = \frac{1}{m} \quad \rightarrow \quad b_1 = \frac{h}{m}$$

$$\frac{b_2}{m-h} = \frac{1}{m} \quad \rightarrow \quad b_2 = \frac{m-h}{m}$$

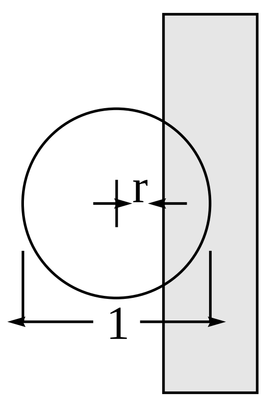
$$A_1 = \frac{1}{2}\frac{h^2}{m}$$

$$A_2 = \frac{1}{2}\frac{(m-h)^2}{m}$$

Square Pixel Filter



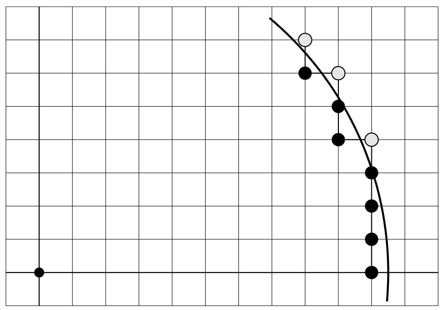
Circular Pixel Filter



$$A(r) = \frac{4}{\pi}\int_{-\frac{1}{2}}^{r}\sqrt{1-4x^2}\,dx$$
$$= \frac{1}{2}+\frac{1}{\pi}\left(2r\sqrt{1-4r^2}+\sin^{-1}2r\right)$$

- Store A in table
- Combine with filter function

Circle Drawing



```
while( n-- ) {
  point(x,y);
  move up;
  if( outside circle )
    move left;
```


Sector

```
sector(int r)
{
  int x = r;
  int y = 0;
  e = -r;

  while( x > y ) {
    point(x,y);
    y += 1; e += 2*y+1;
    if( e >= 0 ) {
      x -= 1; e += 2*x+2;
    }
  }
}
```

$$f(x,y) = (x-\frac{1}{2})^2 + y^2 - r^2 = 0$$

$$f(x,y+1) - f(x,y) = (y+1)^2 - y^2$$
$$= 2y+1$$

$$f(x-1,y) - f(x,y) = (x-\frac{3}{2})^2 - (x-\frac{1}{2})^2$$
$$= -2x+2$$