Computer Graphics

Week 2

Lecture 1

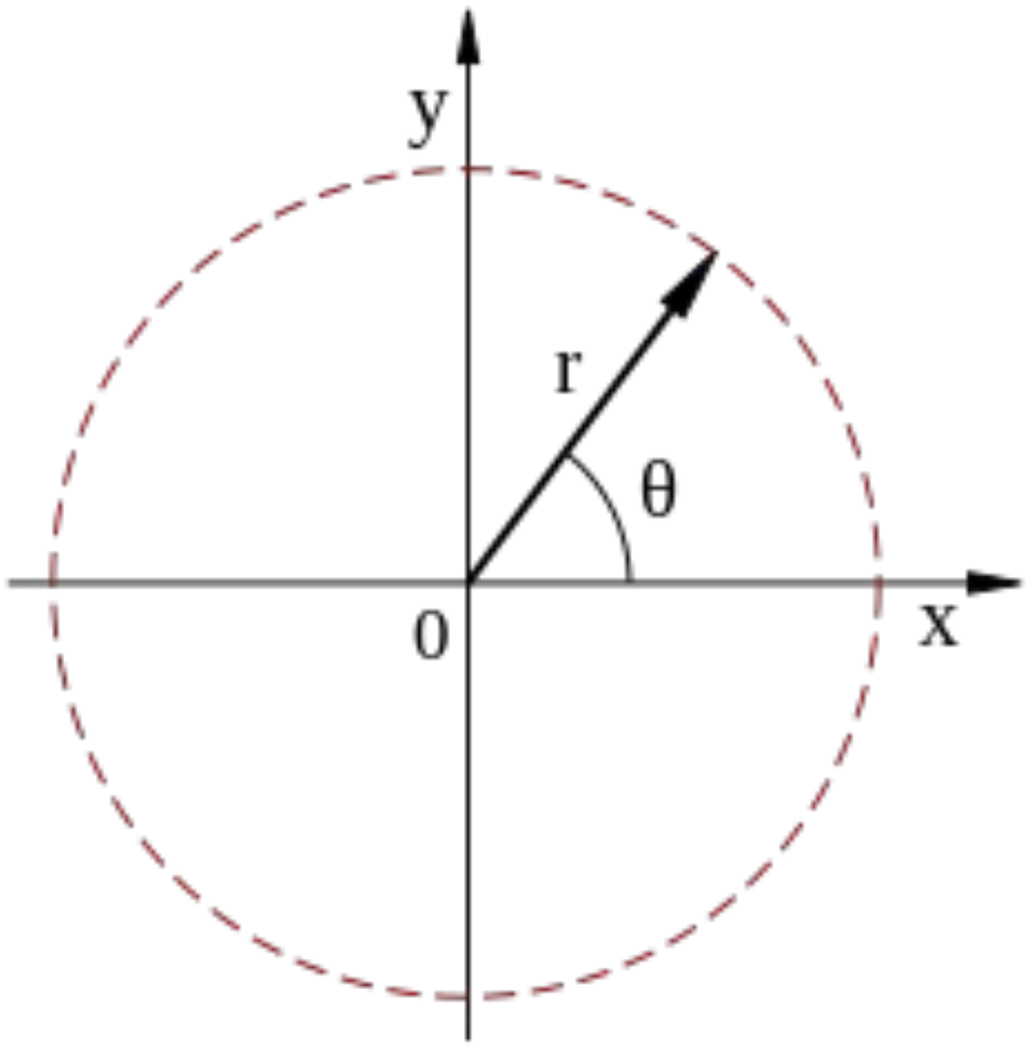
Circle Drawing ALGORITHM

Basics

- **<u>Cartesian Equation:</u>**
  $x^2+y^2=r^2$
  or
  $f(x,y)= x^2+y^2-r^2 = 0$
- **<u>In terms of trigonometric functions:</u>**

  $x=r \sin \theta$
  $y=r \cos \theta$

  for $0^0<=\theta<360^0$

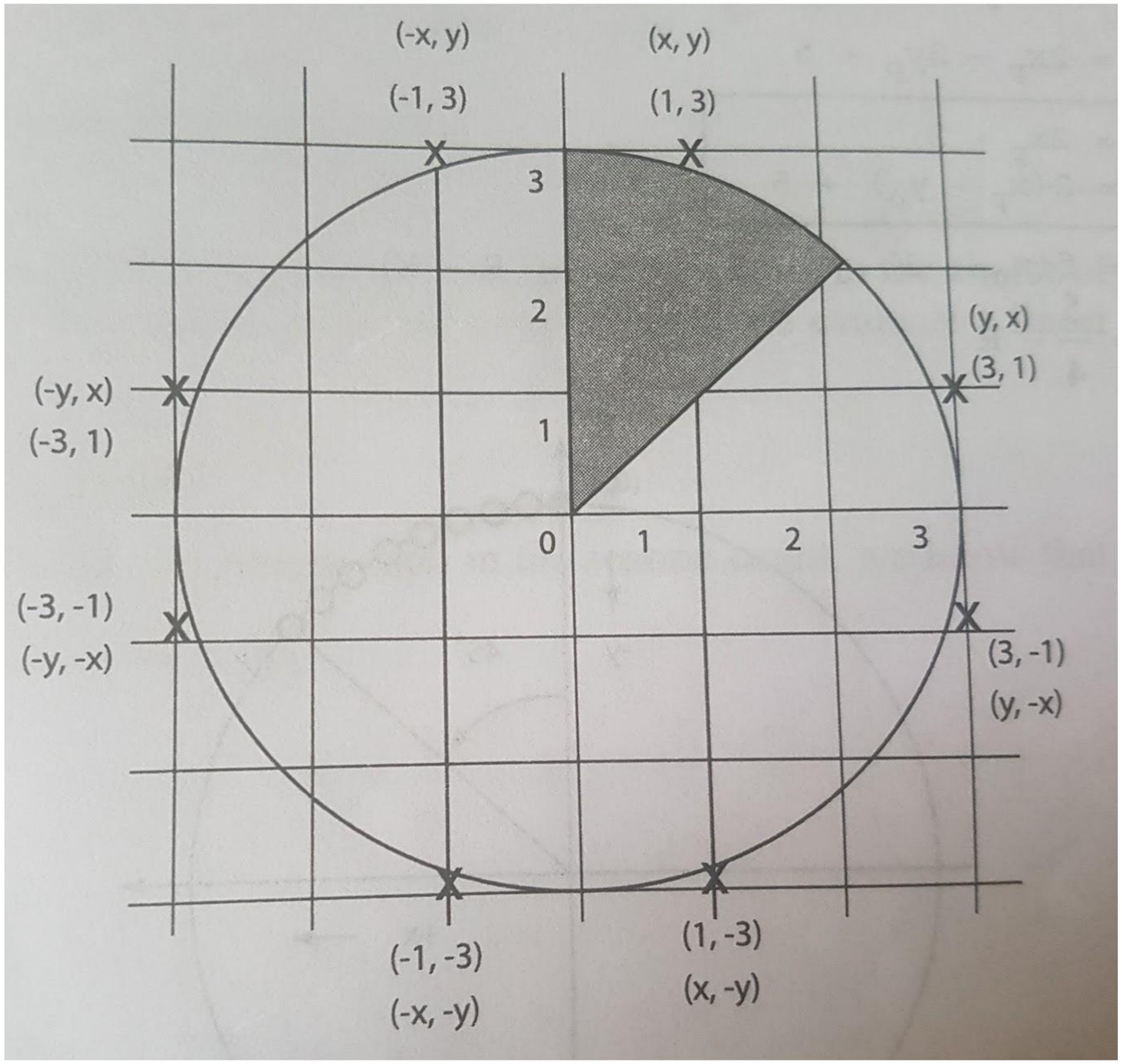


Basics

- $F(x1,y1) = x1^2+y1^2-r^2 = 0 \rightarrow$ Point lies ***on*** the circle
- $F(x2,y2) = x2^2+y2^2-r^2 > 0 \rightarrow$ Point lies ***outside*** the circle
- $F(x3,y3) = x3^2+y3^2-r^2 < 0 \rightarrow$ Point lies ***inside*** the circle

8 Way Symmetry

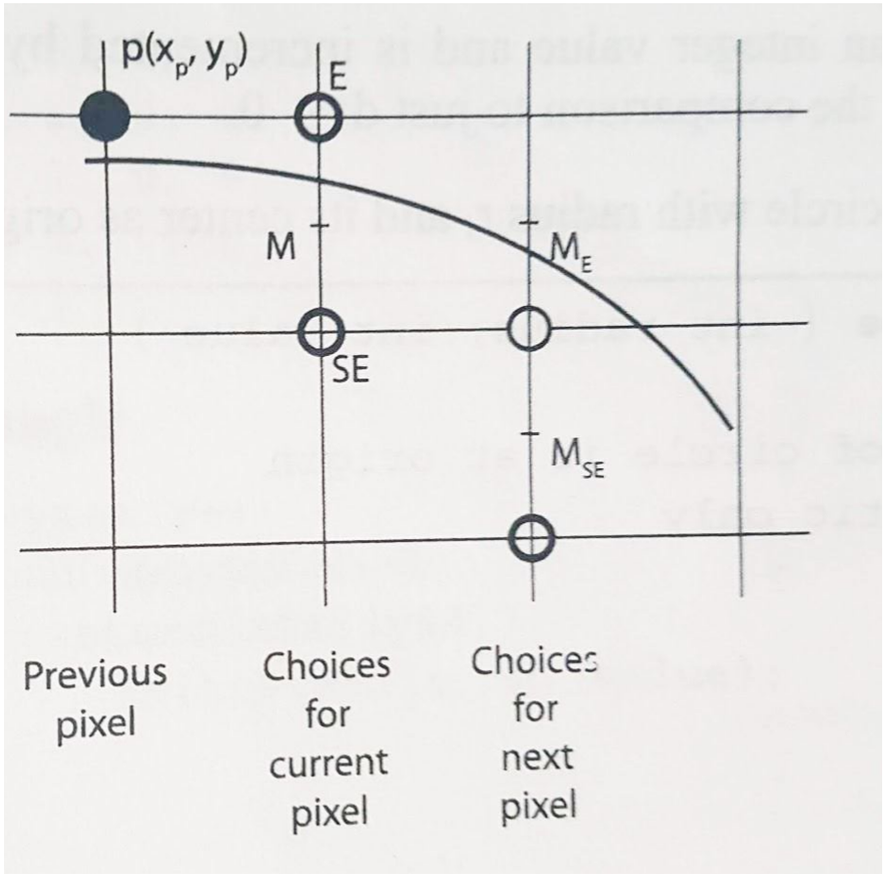
In circle if we just draw points in **one octant** we can get points in **rest of the octants** by **<u>suitable reflections</u>**



8 Way Symmetry

```
void CirclePoints (float x, float y, int value)
{
    WritePixel (  x,  y, value );
    WritePixel (  y,  x, value );
    WritePixel (  y, -x, value );
    WritePixel (  x, -y, value );
    WritePixel ( -x, -y, value );
    WritePixel ( -y, -x, value );
    WritePixel ( -y,  x, value );
    WritePixel ( -x,  y, value );
}
```

Bresenham Algorithm



- We will calculate in just 2nd octant
- Assume starting pixel P is on y-axis
- To find next pixel , see if M is inside or outside of circle

if d <=0 choose E , else choose SE

deltaE $= 2x_p + 3$

deltaSE $=2(x_p-yp)+5$

d_{start}

- In start x_p=0 and y_p=R (radius)
- f(M) = f(xp+1, yp - 1/2) = f(0+1,R-1/2) = d_{start}
- d_{start} = 1-R

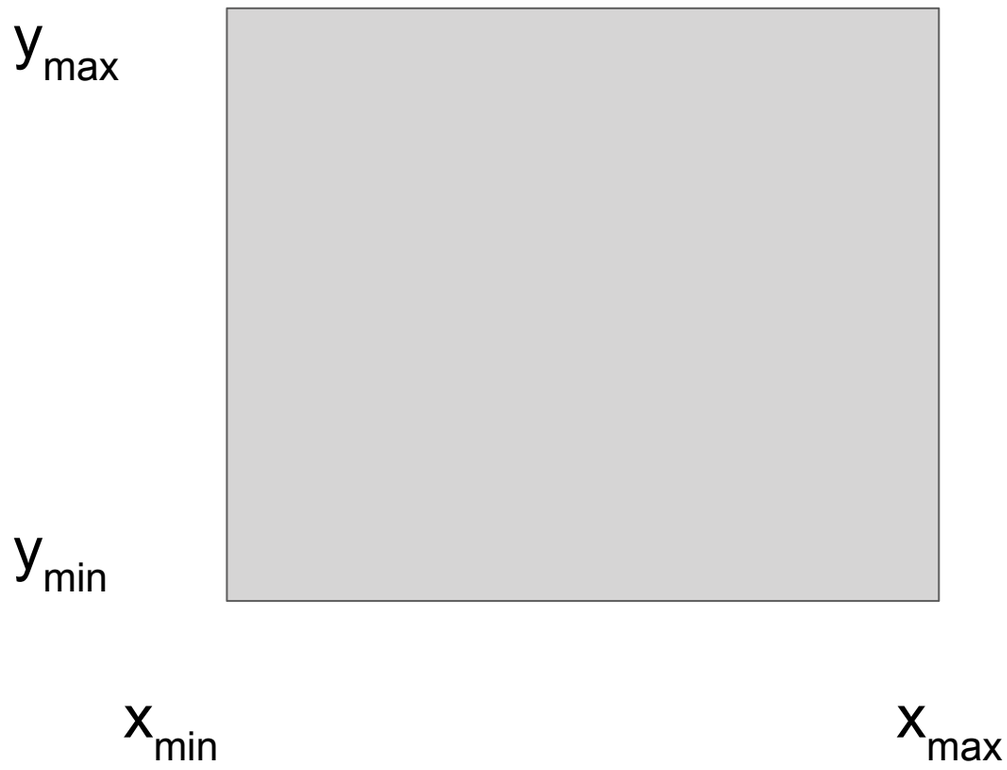
Pseudo Code

```
void MidpointCircle ( int radius, int value )
{
// Assumes center of circle is at origin
// Integer Arithmetic only
   int   x, y;
   int d;
   x = 0;
   y = radius;
   d = 1 - radius;
   CirclePoints ( x, y, value );

   while ( y > x ) {
      if ( d < 0 ) {
         d += x * 2 + 3; // Select East pixel
         x ++;
      }
      else {
         d += (x - y) * 2 + 5;
         x ++;
         y --;
      }
      CirclePoints ( x, y, value );
   }
}
```

Filling Rectangles

```
for (ymin to ymax)
        for (xmin to xmas)
            WritePixel(x,y,value)
```

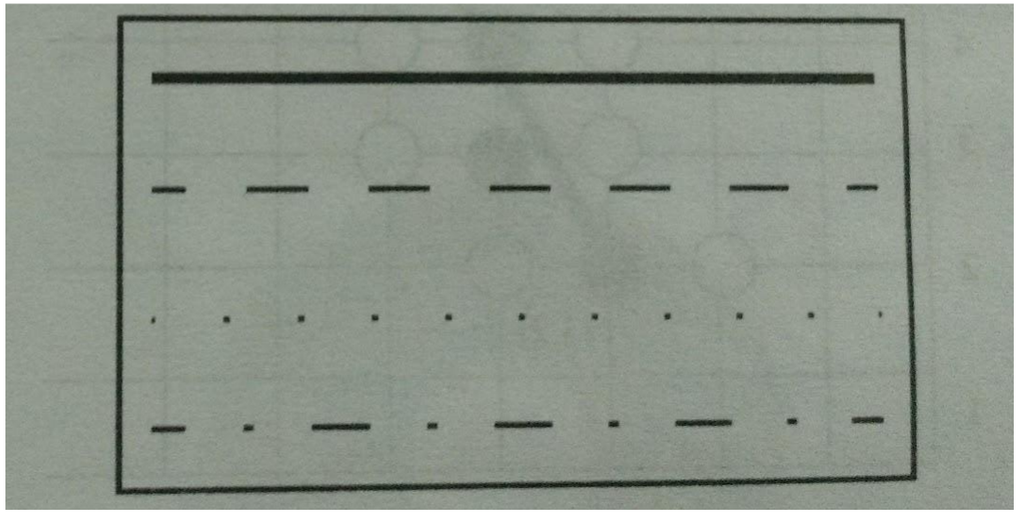


Pattern Filling

```
pattern[4][4]= { 1, 1, 1, 1,
                 1, 0, 0, 1,
                 1, 0, 0, 1,
                 1, 0, 0, 1
                 }
```

```
for(y = ymin; y<ymax;y++)
        for(x = xmin;x<xmax;x++)
                if(pattern[x%4][y%4])
                        WritePixel(x, y, value);
```

Line Styles



- Define the line style as a 16-bit , bit string

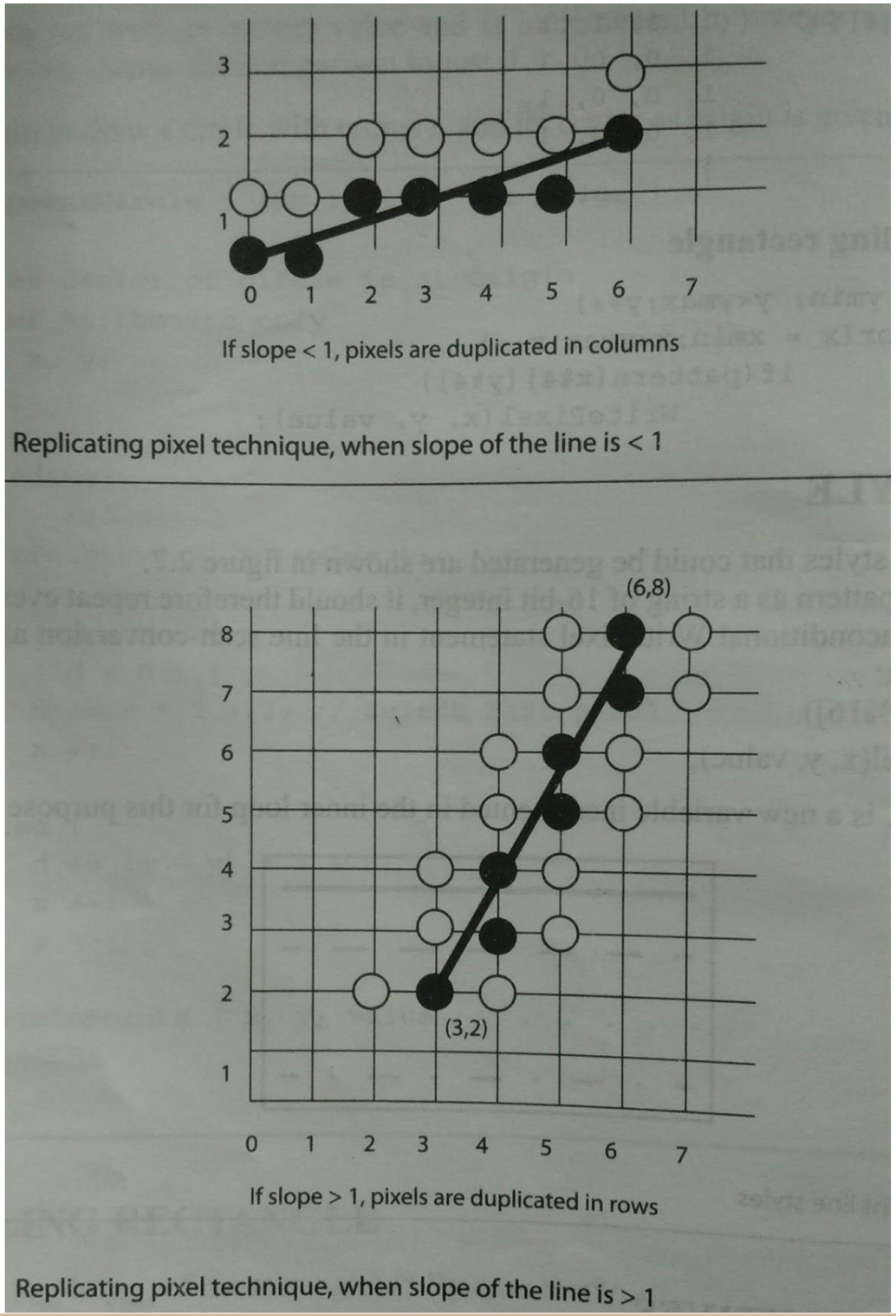
```
if(bitstring[i%16])
    WritePixel(x,y,value);
```

Thickening Primitives

3 techniques are used:

1. Replicating Pixels
2. Moving Pen
3. Filling area between 2 boundaries

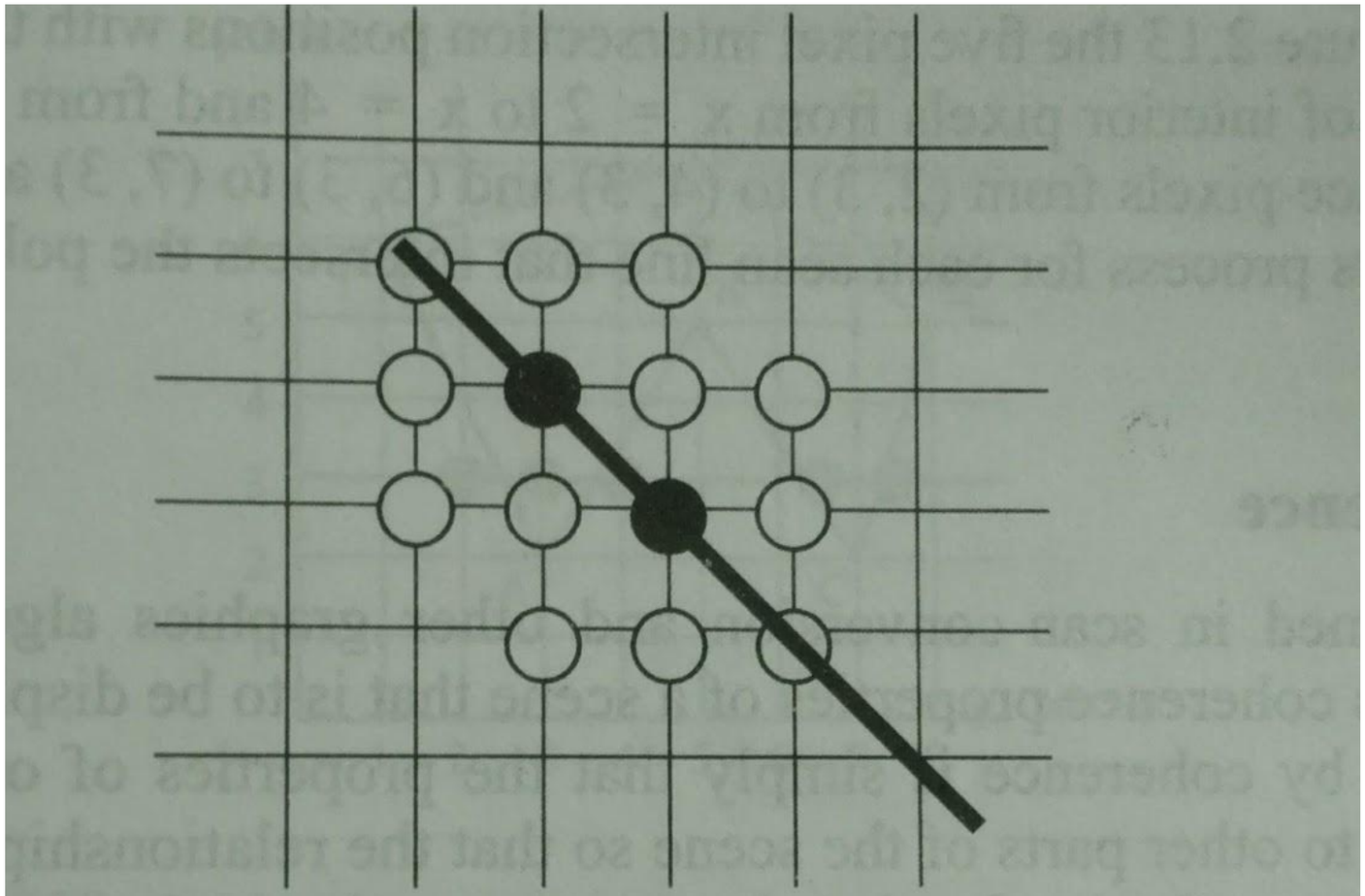
Replicating Pixels



Replicating pixel technique, when slope of the line is < 1

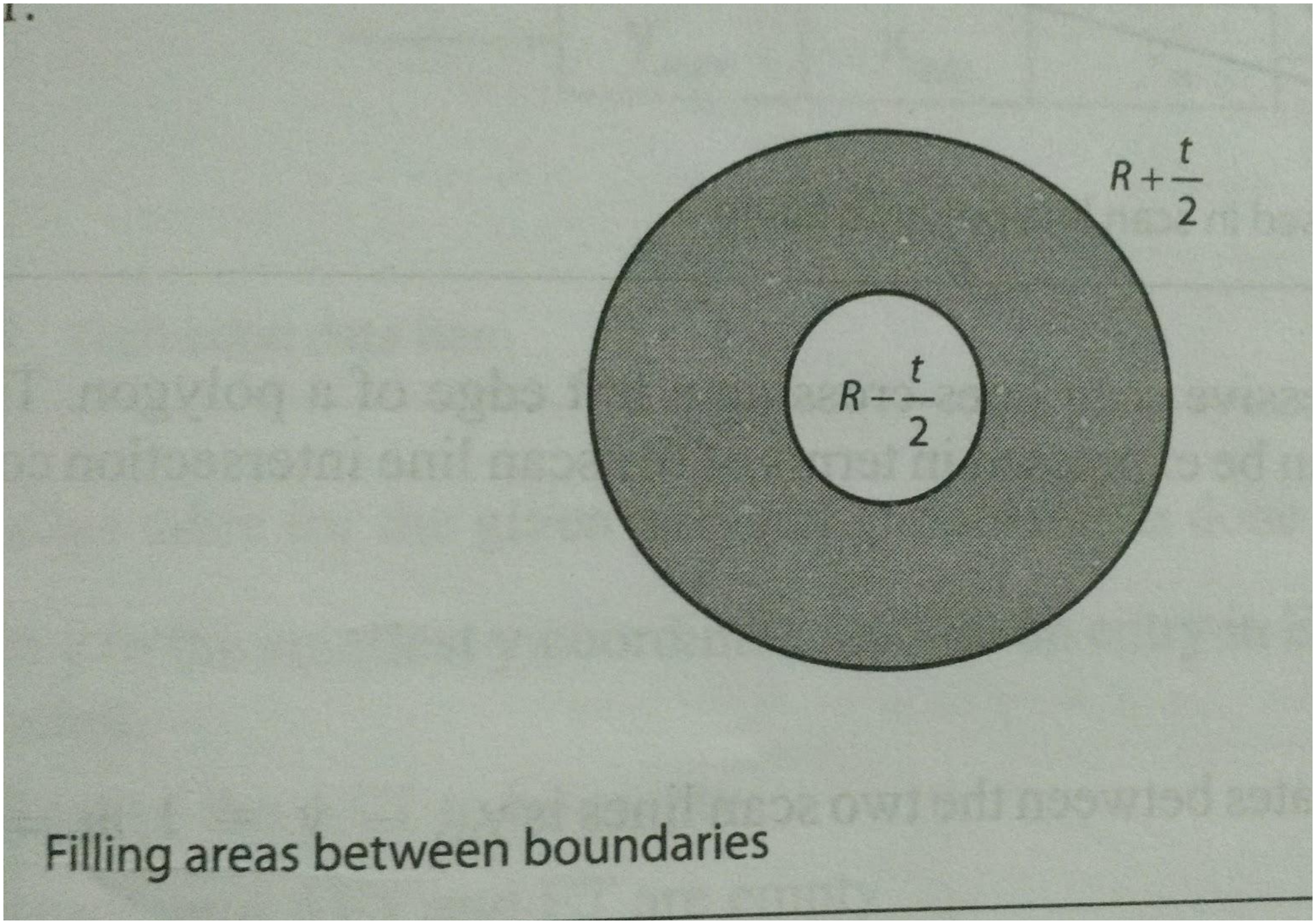
Replicating pixel technique, when slope of the line is > 1

Moving Pen



Center pixel is chosen, all neighbouring 8 pixels are intensified

Filling Area b/w Boundaries



Filling areas between boundaries

The End

Slide 1

Text

TEX

T

[TEXT]

Text

Text

–Johnny Appleseed