

ECE390
Computer Engineering II
Lecture 19



Dr. Zbigniew Kalbarczyk
University of Illinois at Urbana- Champaign

Lecture outline



- Bresenham's Line Drawing Algorithm
- Bresenham's Circle Drawing Algorithm

Drawing Lines Basic Algorithm

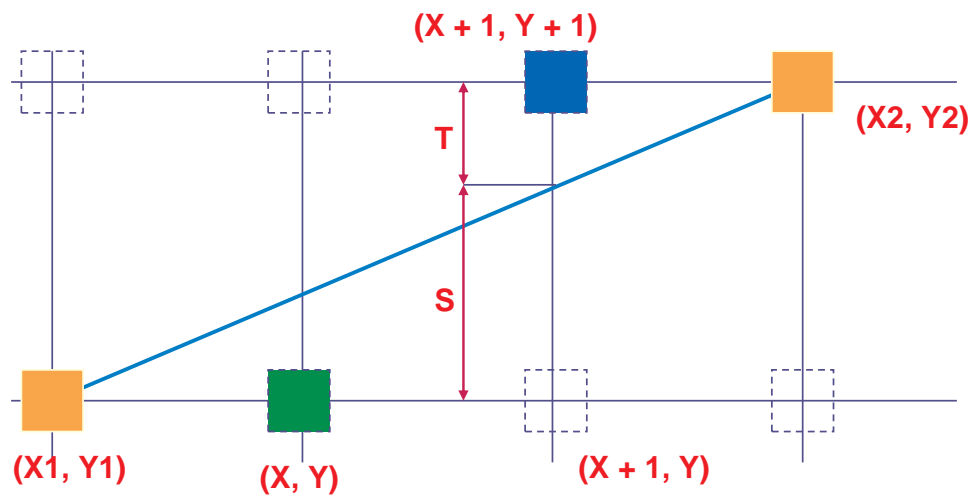
- We want to draw line $Y = m * X + b$
 - m = Slope of line
 - b = Displacement
 - X = independent coordinate
 - Choose X so as to plot larger number of points
 - Y = Dependent coordinate
 - Plot to nearest Y when fractions occur
- Draw Line from $(X1, Y1)$ to $(X2, Y2)$
 - $m = dy/dx$; $dy = (Y2-Y1)$; $dx = (X2-X1)$
 - $b = Y1 - m*X1$
 - Plot($X, m*X+b$) for $X=X1... X2$
- This algorithm requires slow floating point math

Derivation of Bresenham's Line Algorithm

- Eliminates floating point calculations
- Given two line endpoints, we are trying to find the “in-between” points on a pixel grid
- The Bresenham's line algorithm determines subsequent points from the *start point* by making a *decision* between the two next available points and selecting one that is closer to the *ideal point*

- Start point $(X1, Y1)$
- End point $(X2, Y2)$
- Slope $m = dy/dx$
- Current Point (X, Y)

Derivation of Bresenham's Line Algorithm (cont.)



Derivation of Bresenham's Line Algorithm (cont.)

- If we restrict the slope for now to $(0 \leq \text{slope} \leq 1)$ and assume $(X_1 < X_2)$
 - step in x one pixel at a time to the right and determine what y value to choose next
 - given current point (X, Y) , next ideal point would be $(X + 1, Y)$
 - must choose between $(X + 1, Y)$ or $(X + 1, Y + 1)$
- How do we decide between these points?
 - must choose the closet point to the ideal

Derivation of Bresenham's Line Algorithm (cont.)

- Determine S and T

$$S = (X + 1) * m - Y$$

$$T = 1 - S = 1 - (X + 1) * m + Y$$

- Then $S - T = 2 * (X + 1) * m - 2 * Y - 1$
- Select the next point:

if $S < T$ go horizontal (**next point (X + 1, Y)**)

if $S > T$ go diagonal (**next point (X + 1, Y + 1)**)

Derivation of Bresenham's Line Algorithm (cont.)

- We want to develop a **FAST** algorithm; we create a decision variable (error) that can be used to quickly determine which point to use

$$E = (S - T) * dx$$

$$= 2 * (X + 1) * dy - 2 * Y * dx - 1 * dx$$

$$= 2 * X * dy - 2 * Y * dx + 2 * dy - dx$$

NOTE, that $m = dy/dx$; the quantity dx is non-zero positive

$E(i) = E(X, Y)$ value of E at ith point

$$E(0) = E(0, 0) = 2 * dy - dx$$

$E(i+1) - E(i) =$ Additive error at each point

$$\text{Diagonal Move: } E(X+1, Y+1) - E(X, Y) = 2 * dy - 2 * dx$$

$$\text{Horizontal Move: } E(X+1, Y) - E(X, Y) = 2 * dy$$

Algorithm for Drawing Lines in the First Octant

Draw_Line (X1, Y1, X2, Y2)

integer dx = |X2 - X1|

integer dy = |Y2 - Y1|

E = 2dy - dx

HorIncr = 2dy

DiagIncr = 2dy - 2dx

Y = Y1

for X from X1 to X2

set_pixel (X, Y)

if E < 0 then

E = E + HorIncr

else

E = E + DiagIncr

Y = Y + 1

end if

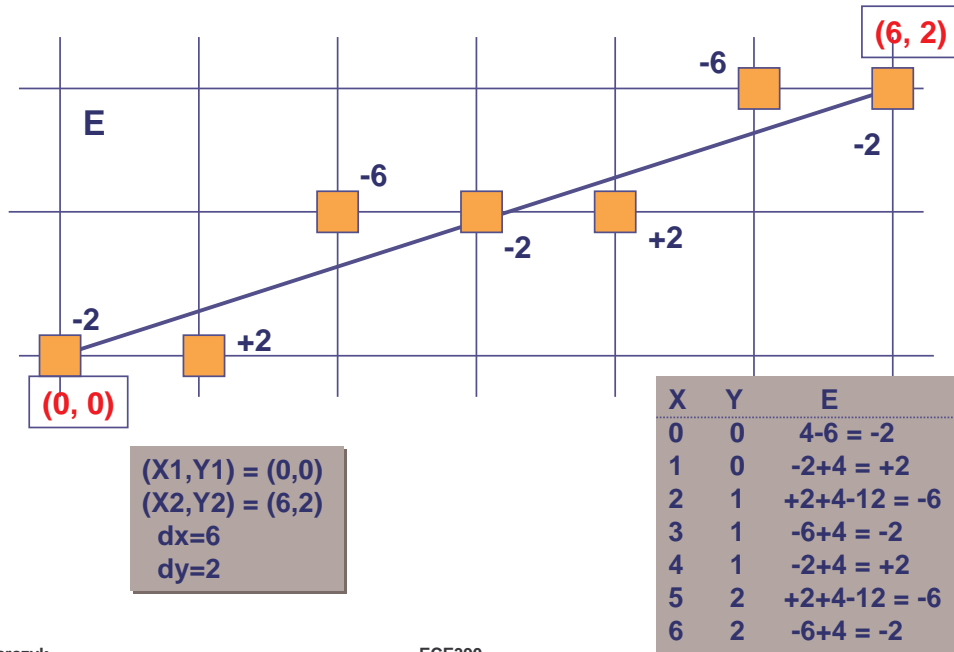
end for

end Draw_Line

Algorithm for Drawing Lines in the First Octant (cont.)

- The code should not use
 - floating point numbers
 - multiplication
 - division
 - comparison to any number other than zero (they are slower)
- It should be sufficient to use
 - additions
 - bit-shifts
 - comparison to zero

Example (first octant)

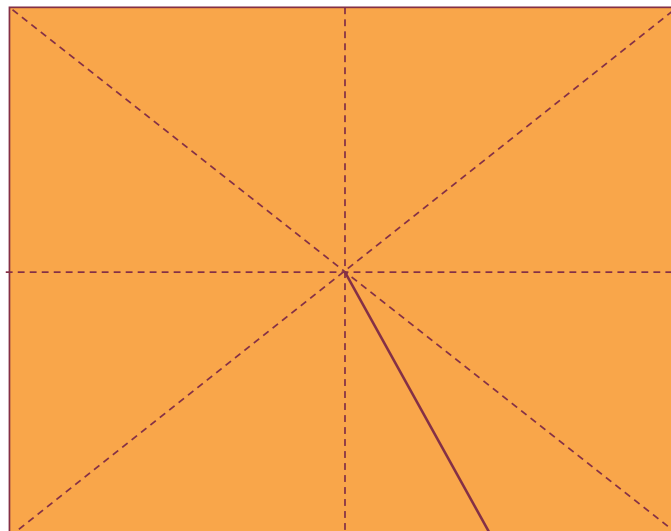


Z. Kalbarczyk

ECE390

11

Example (Seventh Octant)

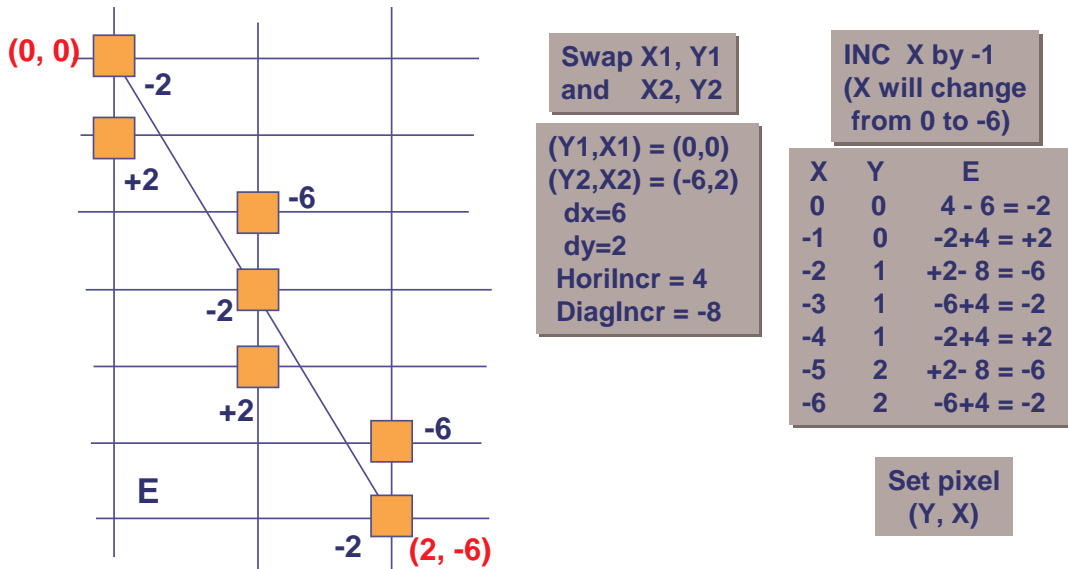


Z. Kalbarczyk

ECE390

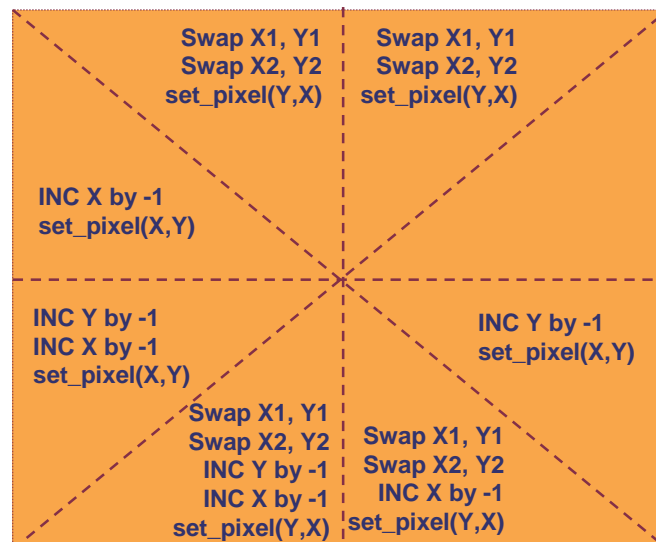
12

Example (Seventh Octant cont.)



Drawing Lines in All Eight Octants

- There are eight regions (octants) in which the algorithm should work



Drawing Circles

- Circles are symmetric about the x and y axes, as well as their 45° lines
- We only have to figure out the pixels for one octant of the circle!
- Bresenham has an algorithm for doing this

Drawing circles

```
e = 3 - (2 * RADIUS)
x = 0
y = RADIUS
Do Until x = y
  PutPixel(CenterX + X, Center Y + Y)
  PutPixel(CenterX + X, Center Y - Y)
  PutPixel(CenterX - X, Center Y + Y)
  PutPixel(CenterX - X, Center Y - Y)
  PutPixel(CenterX + Y, Center Y + X)
  PutPixel(CenterX + Y, Center Y - X)
  PutPixel(CenterX - Y, Center Y + X)
  PutPixel(CenterX - Y, Center Y - X)

  if e < 0 then
    e = e + (4 * x) + 6
  else
    e = e + 4 * (x - y) + 10
    y = y - 1
  end
  x = x + 1
End do
```

