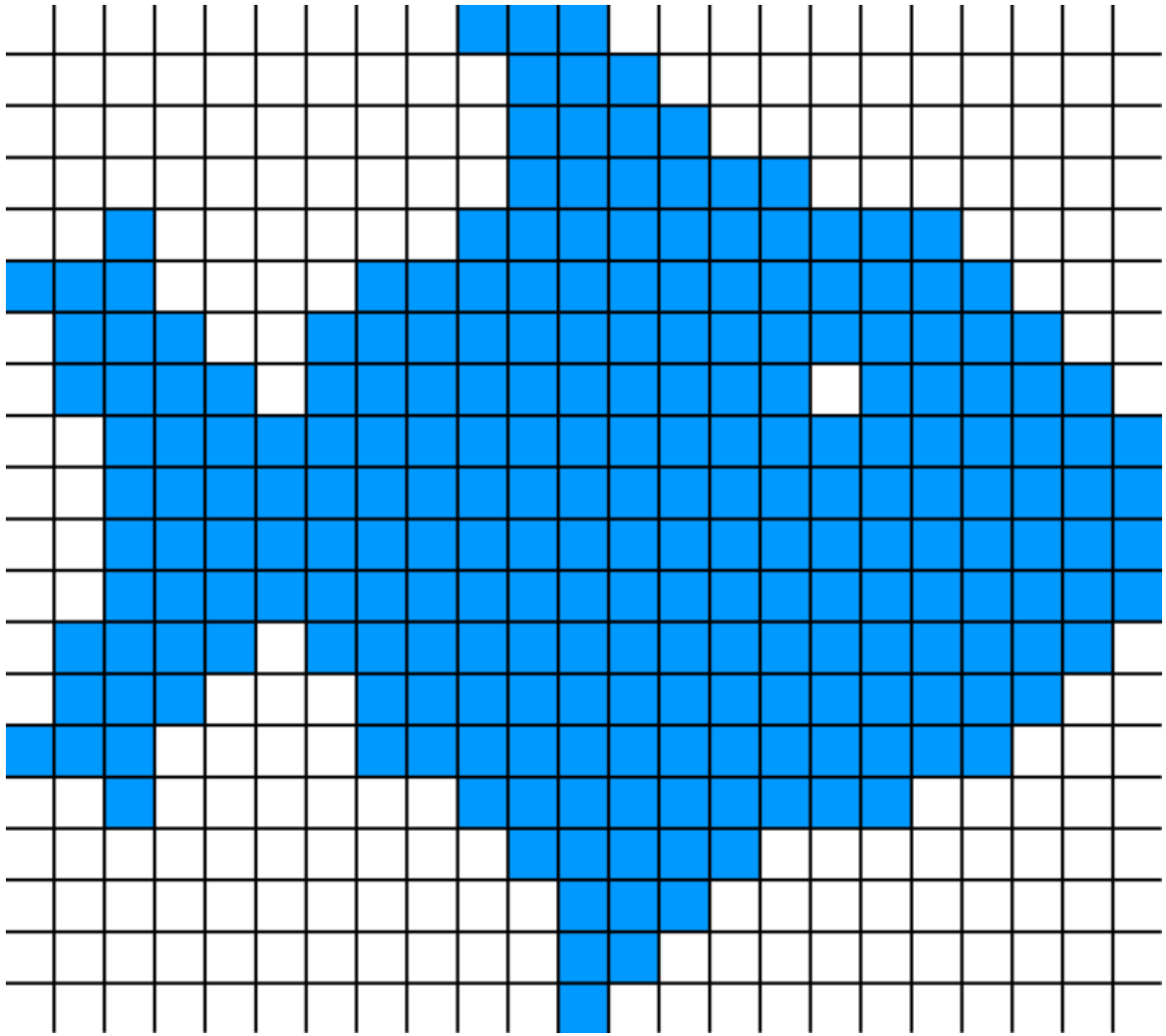
CS 461 - Computer Graphics

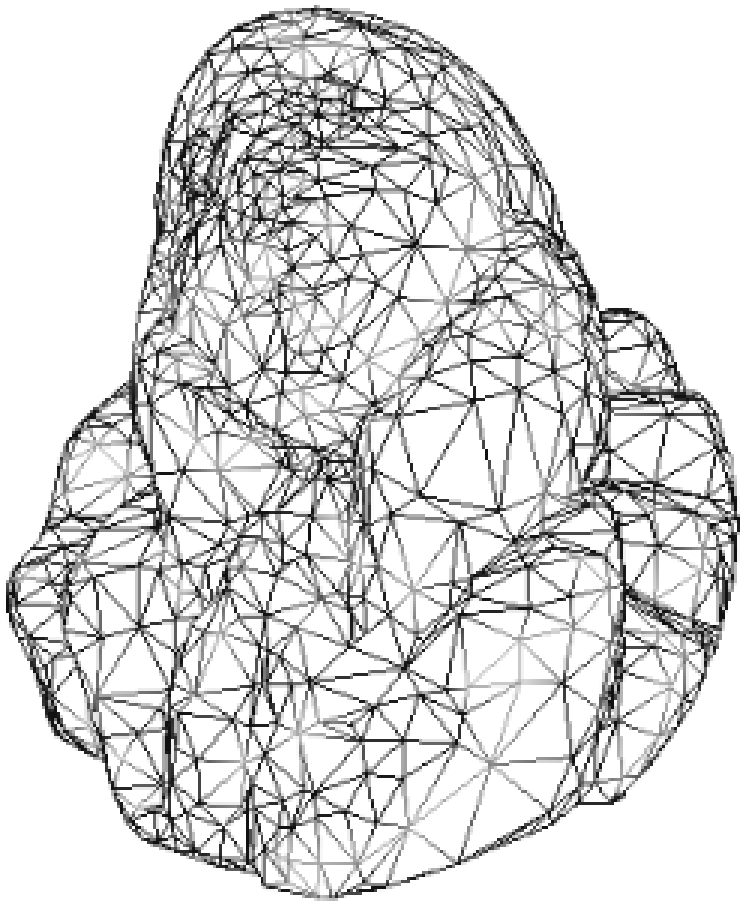
Drawing basic primitives - I

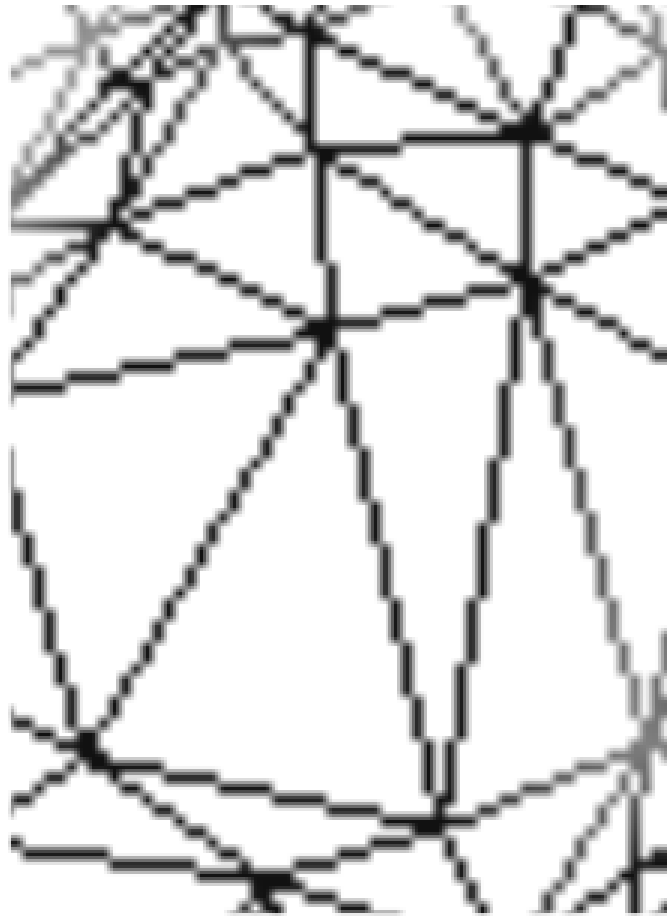
Amal Dev Parakkat

Raster image



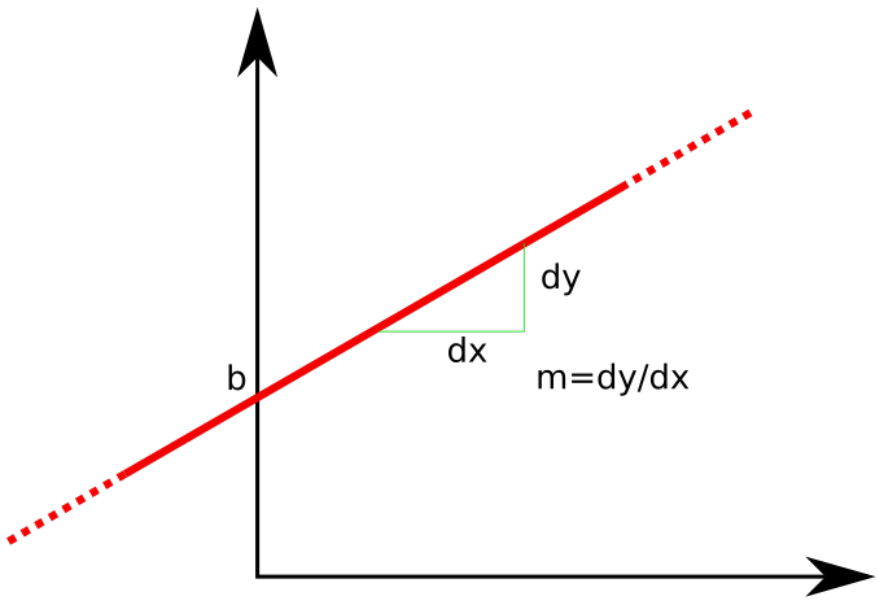
Stair-step (Jaggies)

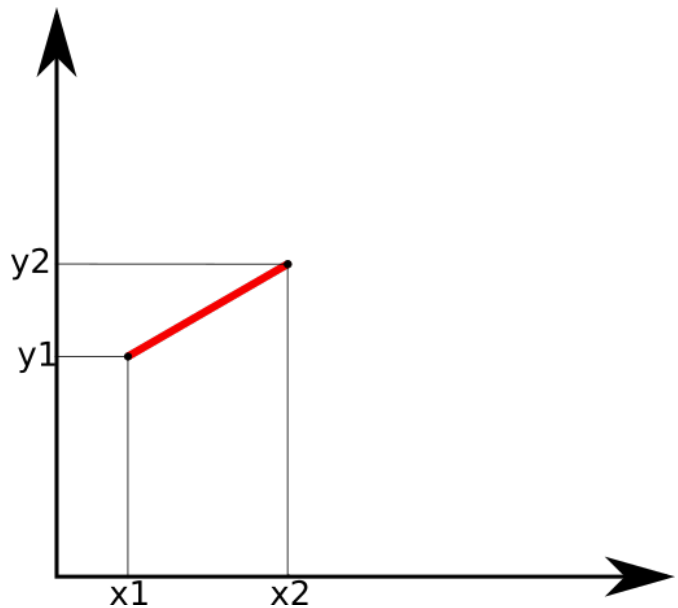




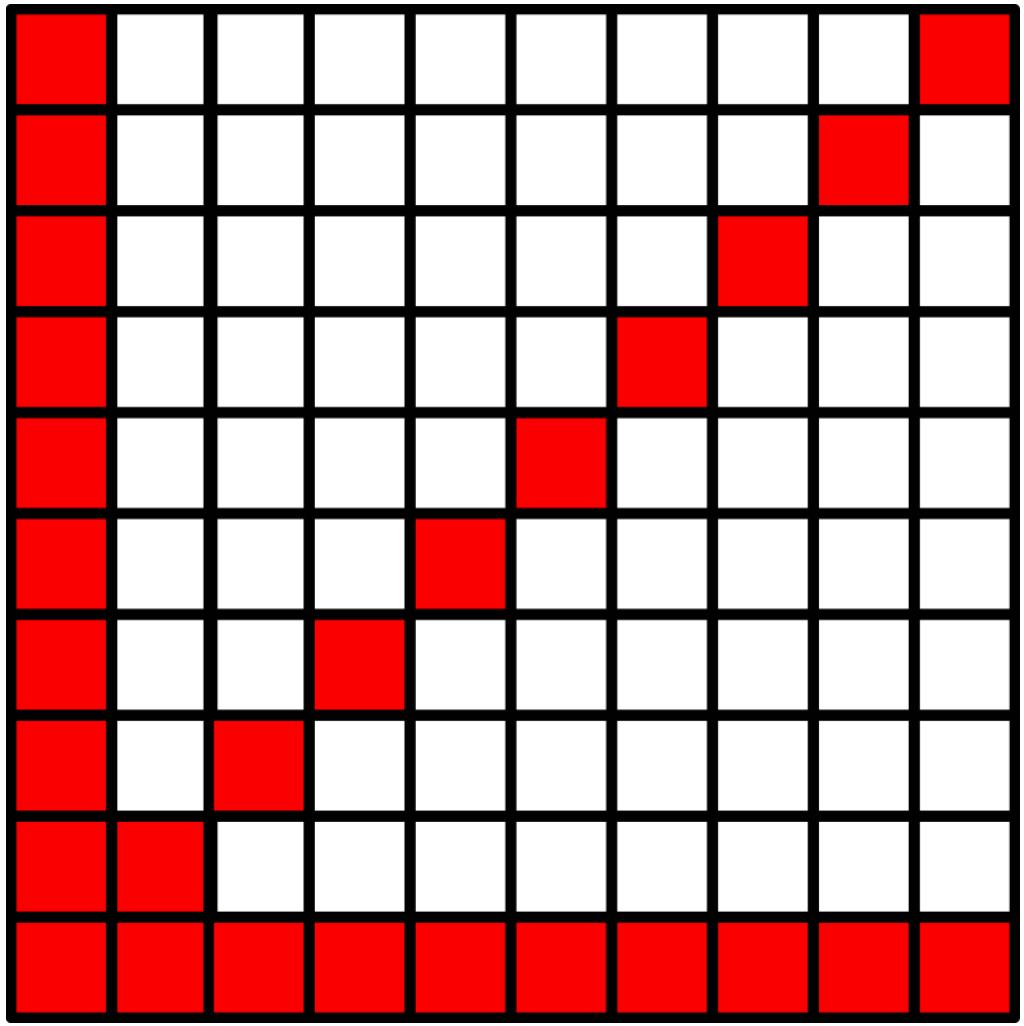
Line equation

$$y = m.x + b \tag{1}$$

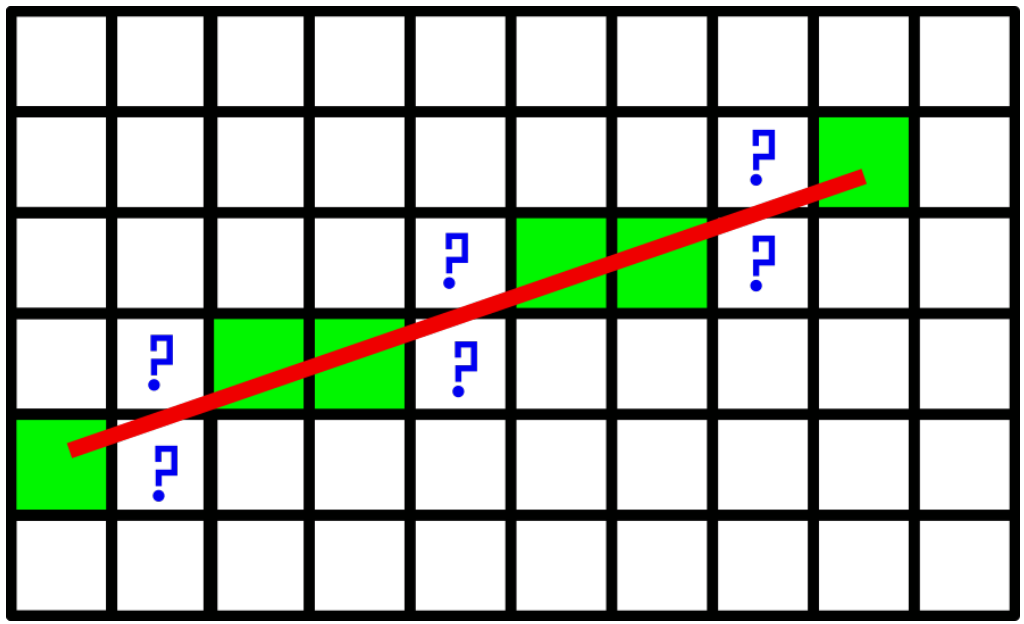




Best cases



Confusion!!!



Algorithms

- Digital Differential Analyzer (DDA)
- Bresenham's algorithm

DDA

- Simple algorithm
- Sample the line segment into small pieces and increment in an iterative fashion
- Let dx and dy be the horizontal and vertical distance between endpoints
- Number of steps = max(dx,dy)
- Divide corresponding x-coordinates into $\frac{dx}{steps}$ pieces
- Divide corresponding y-coordinates into $\frac{dy}{steps}$ pieces
- Starting from x=x0 and y=y0, repeatedly do $x + \frac{dx}{steps}$ and $y + \frac{dy}{steps}$

DDA Algorithm

Algorithm 1: Line drawing using DDA

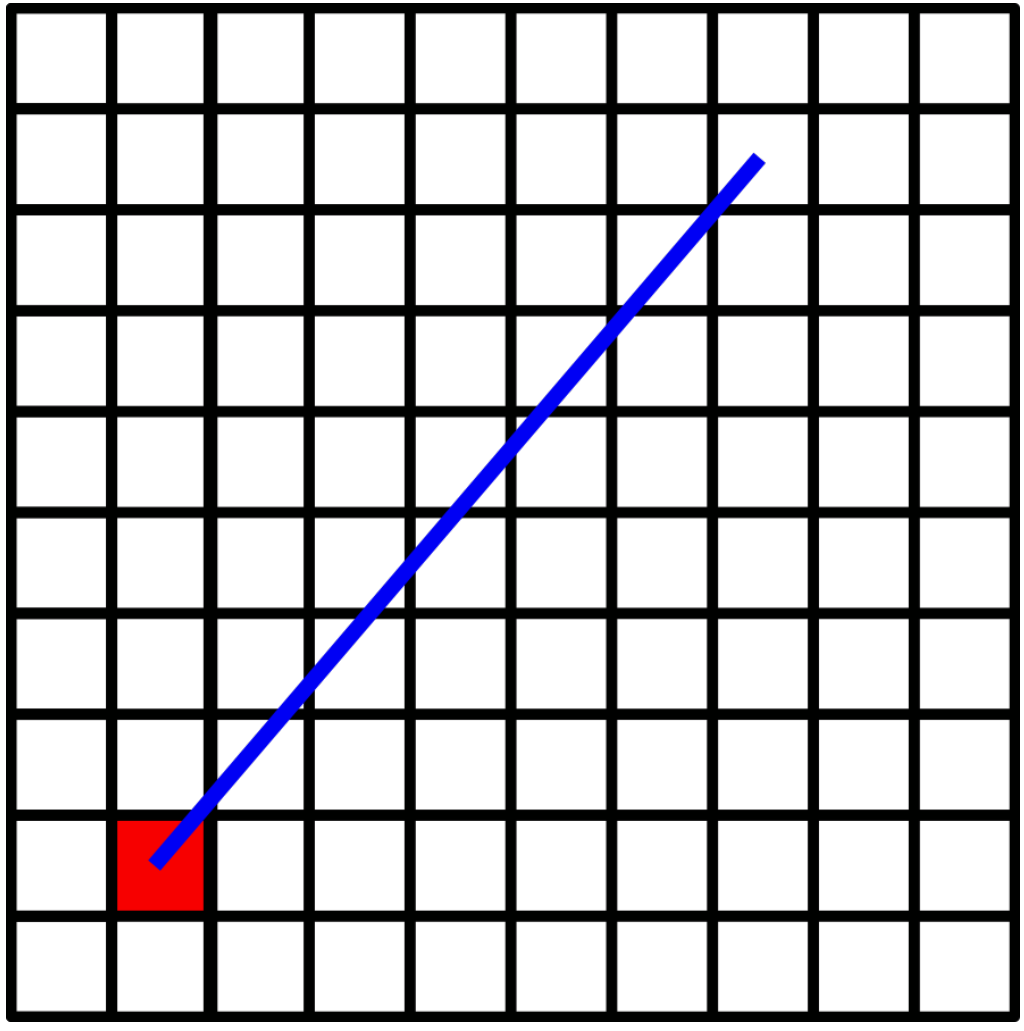
```
Set dx=x1-x0 and dy=y1-y0
Set x=x0 and y=y0
if abs(dx)>abs(dy) then
  |   steps=abs(dx)
else
  |   steps=abs(dy)
end
Set xinc=dx/steps and yinc=dy/steps
SetPixel(round(x),round(y))
for k=0 to steps do
  |   x=x+xinc and y=y+yinc
  |   SetPixel(round(x),round(y))
end
```

DDA Example

- Line segment from (2,2) to (8,9)
- dx=6, dy=7
- steps=7
- xinc=6/7, yinc=1

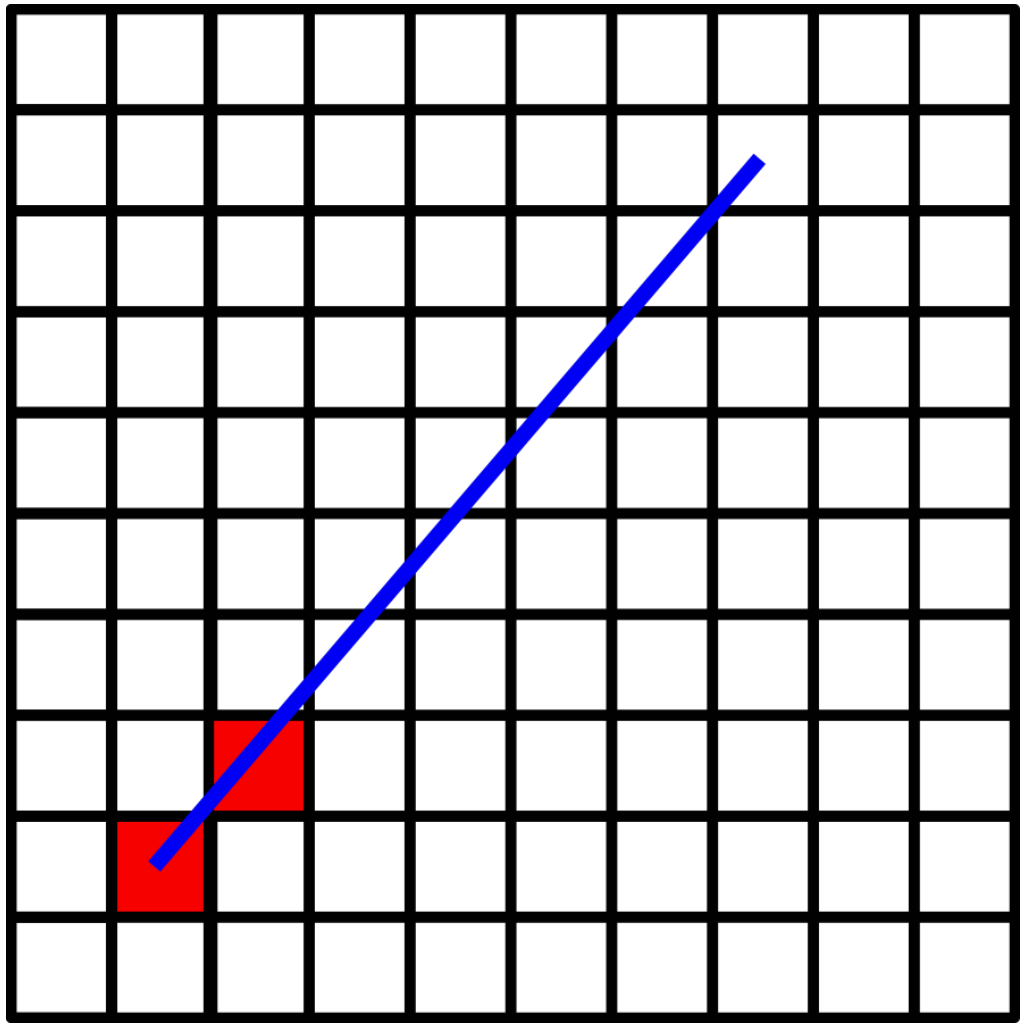
DDA Example

- x=2,y=2



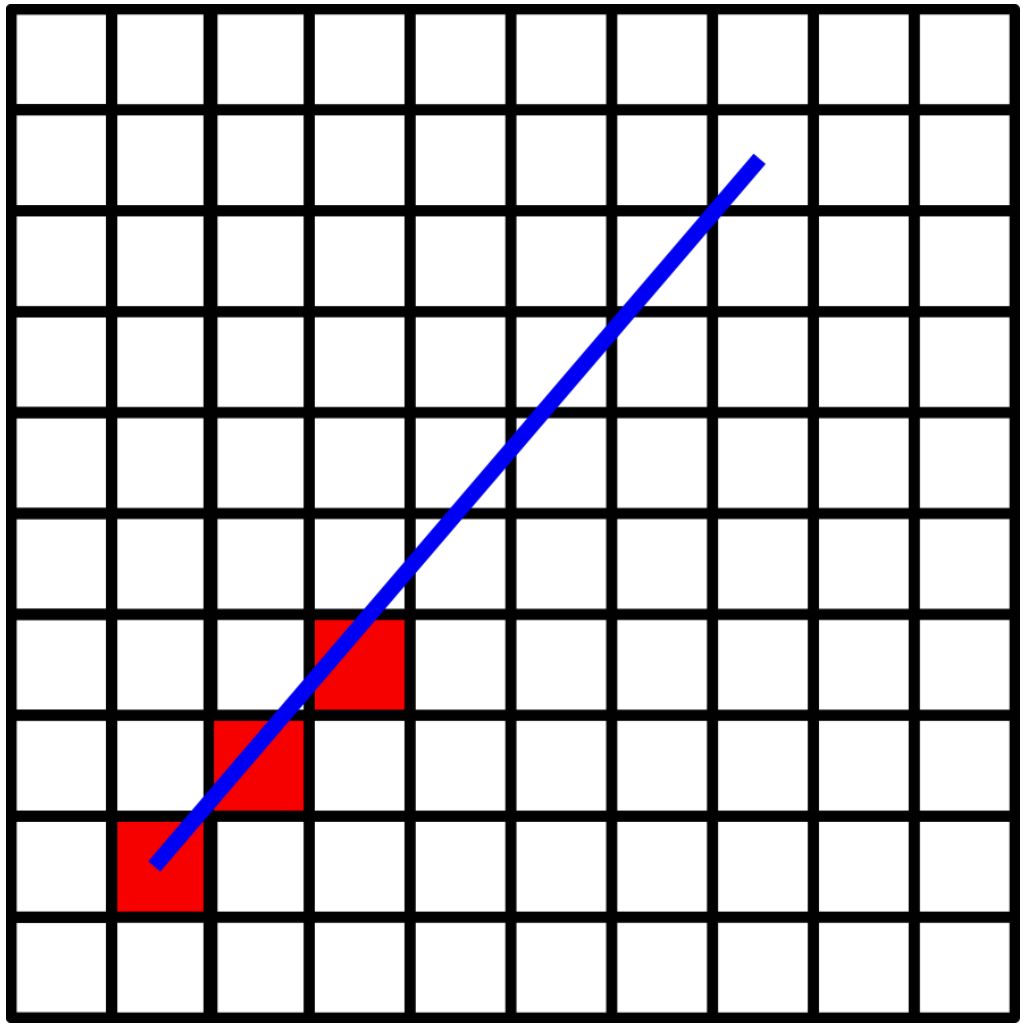
DDA Example

- x=2+(0.8571)=2.8571≈3, y=2+1=3



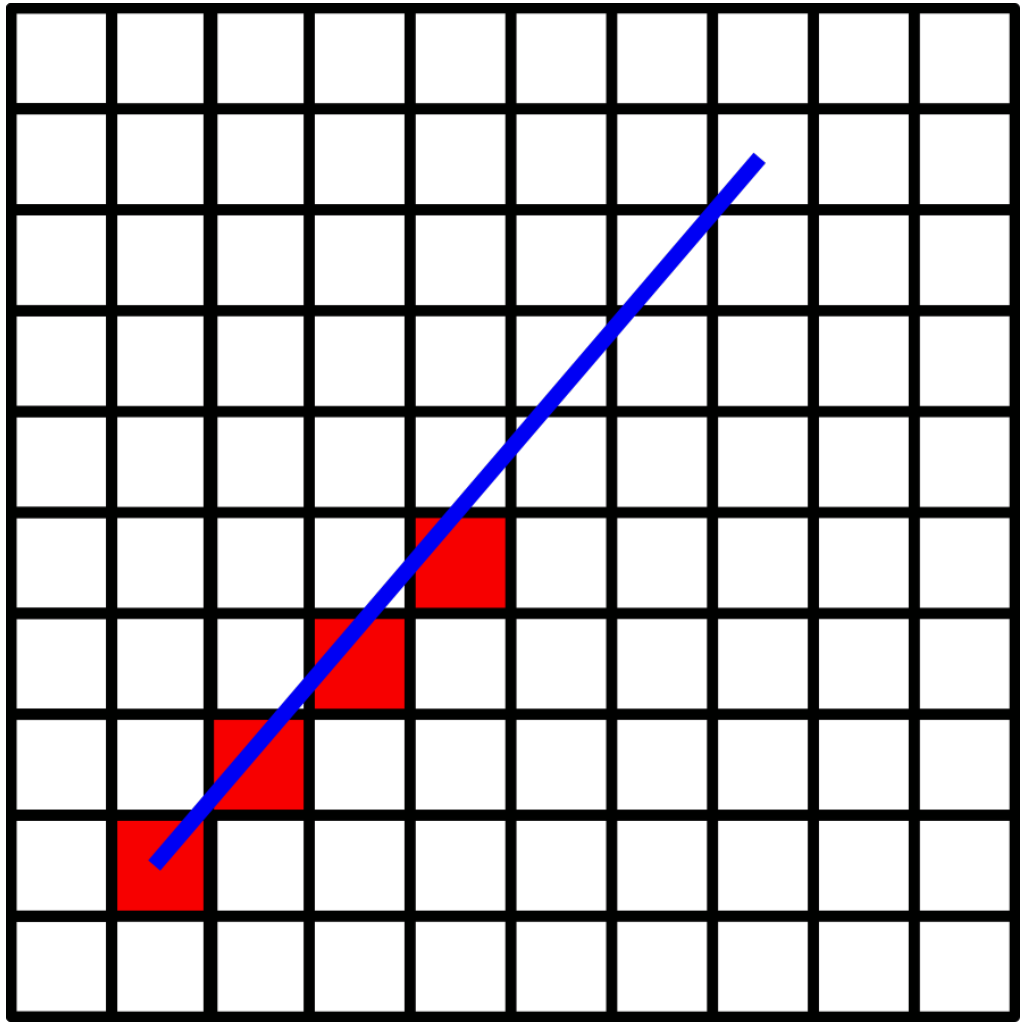
DDA Example

- x=2.8571+(0.8571)=3.7142≈4, y=3+1=4



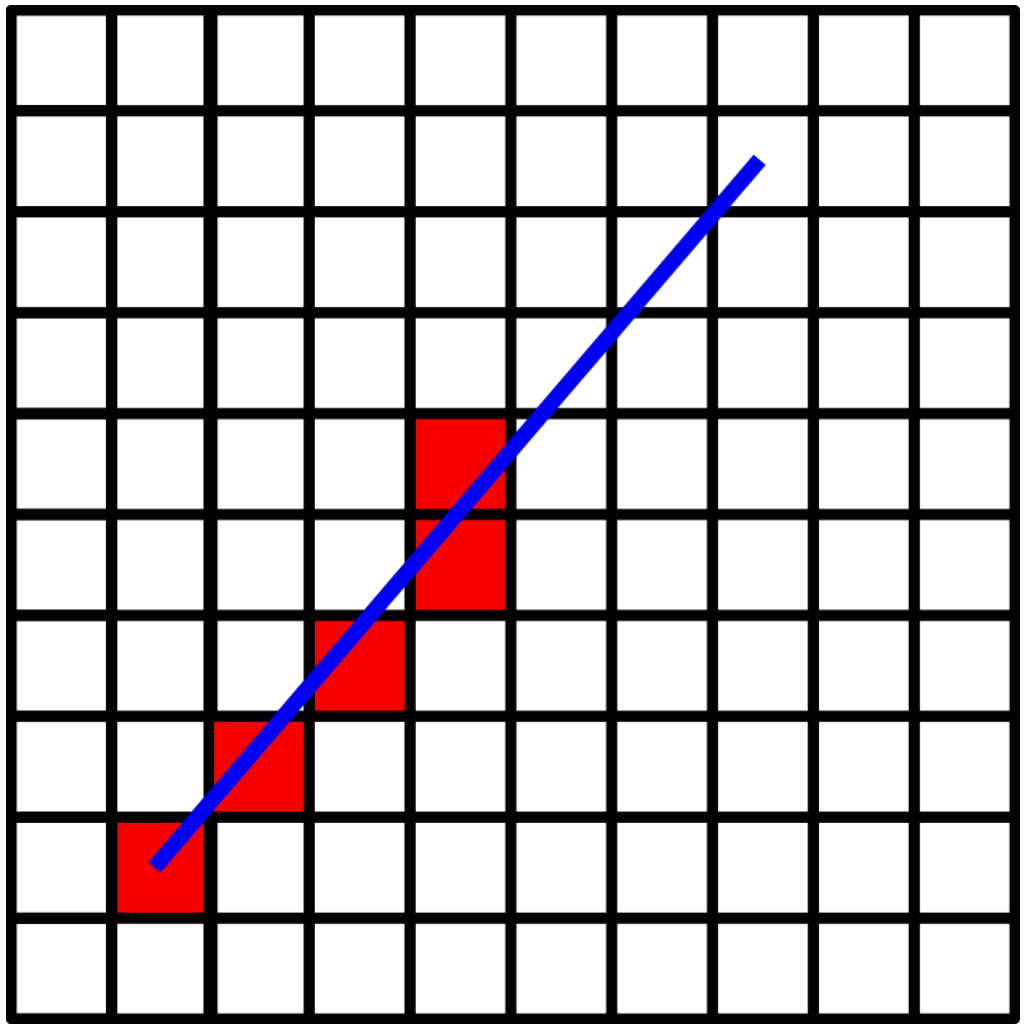
DDA Example

- x=3.7142+(0.8571)=4.5713≈5, y=4+1=5



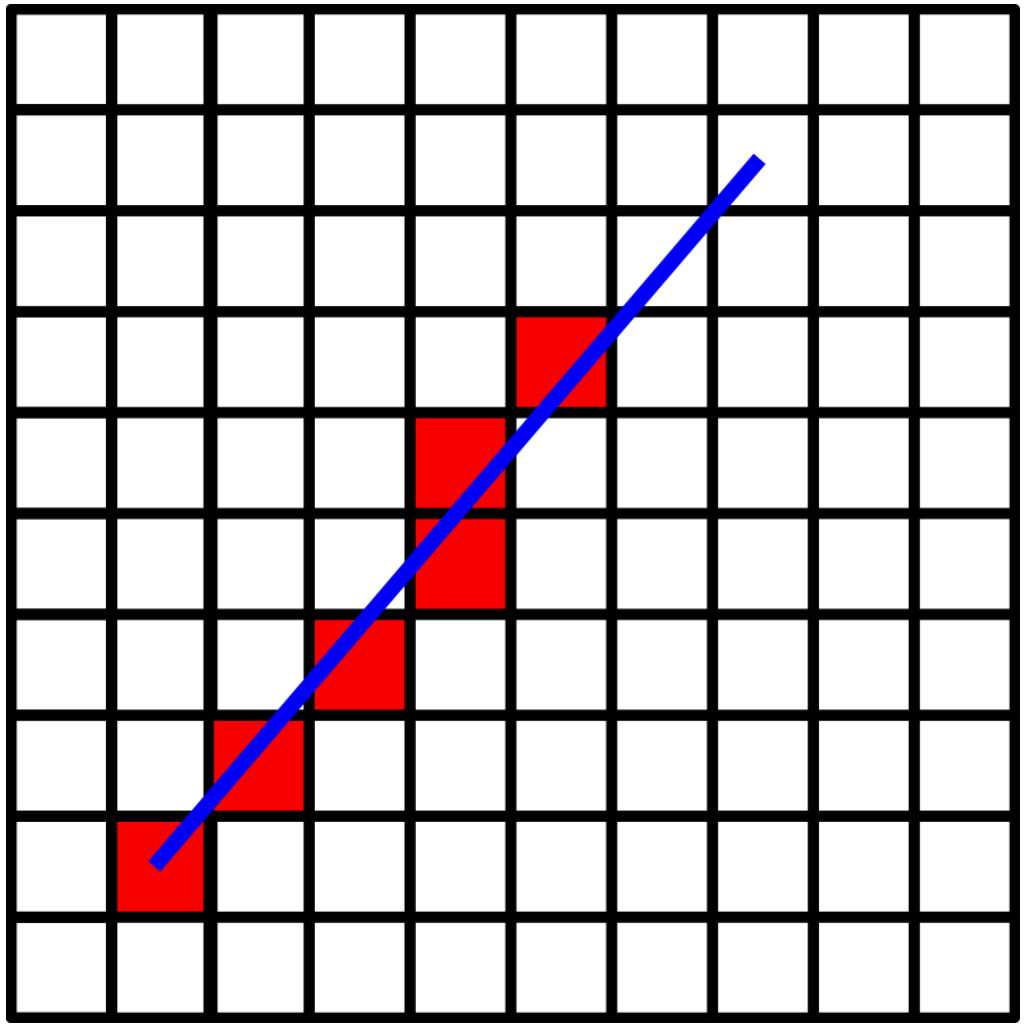
DDA Example

- x=4.5713+(0.8571)=5.4284≈5, y=5+1=6



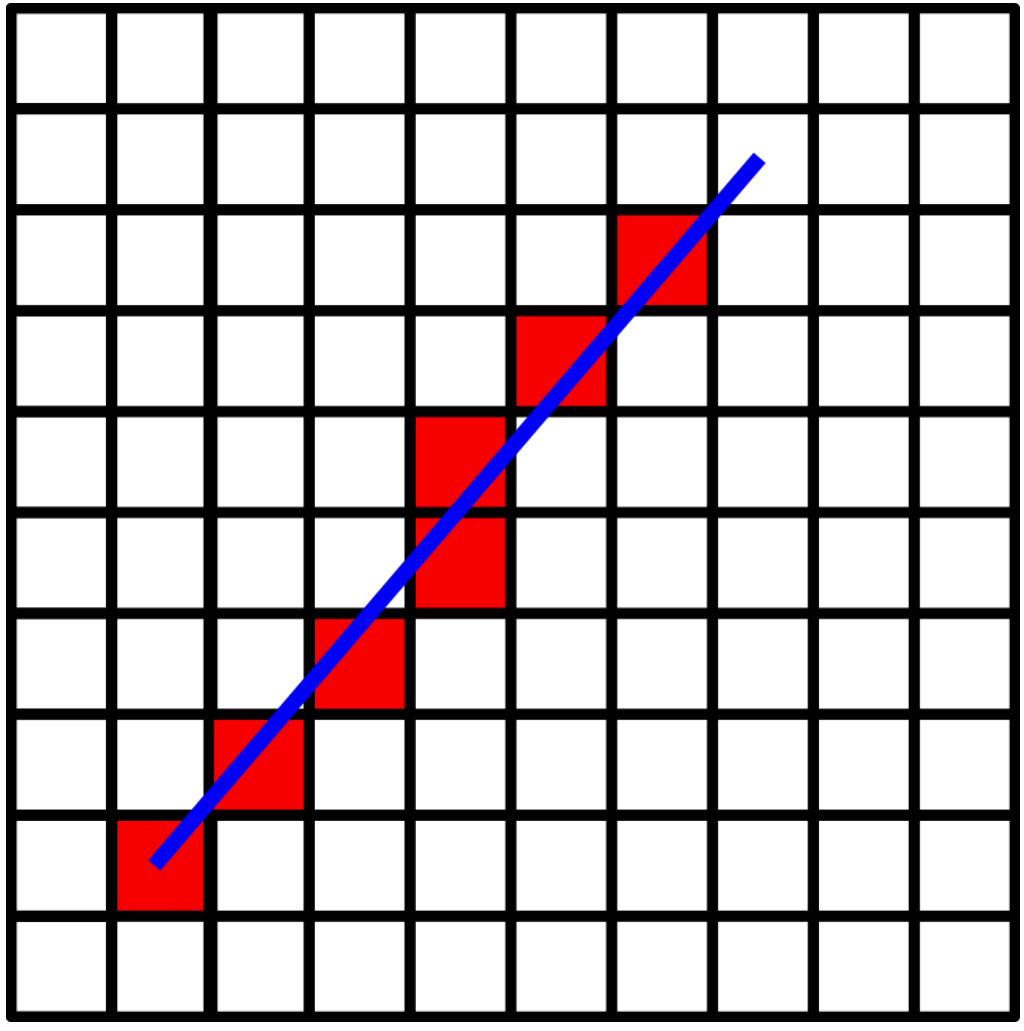
DDA Example

- x=5.4284+(0.8571)=6.2855≈6, y=6+1=7



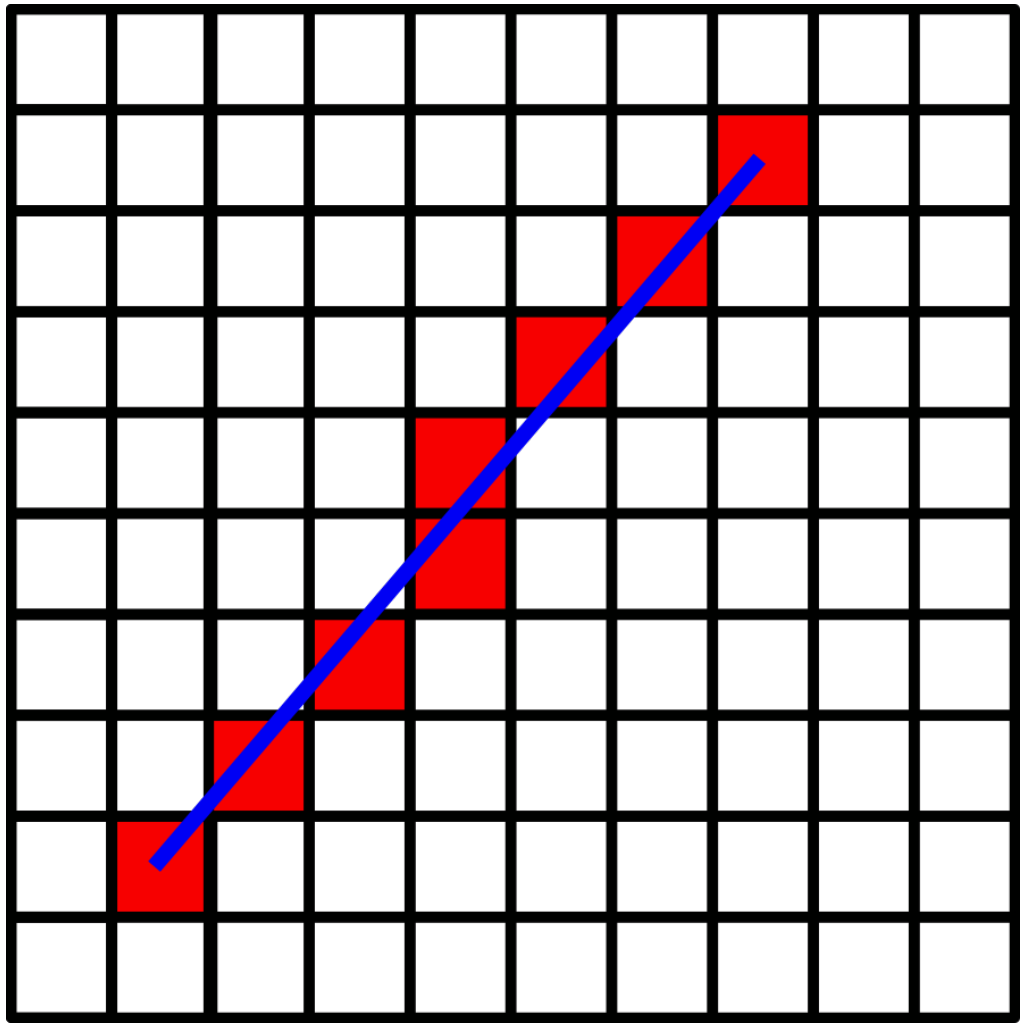
DDA Example

- x=6.2855+(0.8571)=7.1426≈7, y=7+1=8



DDA Example

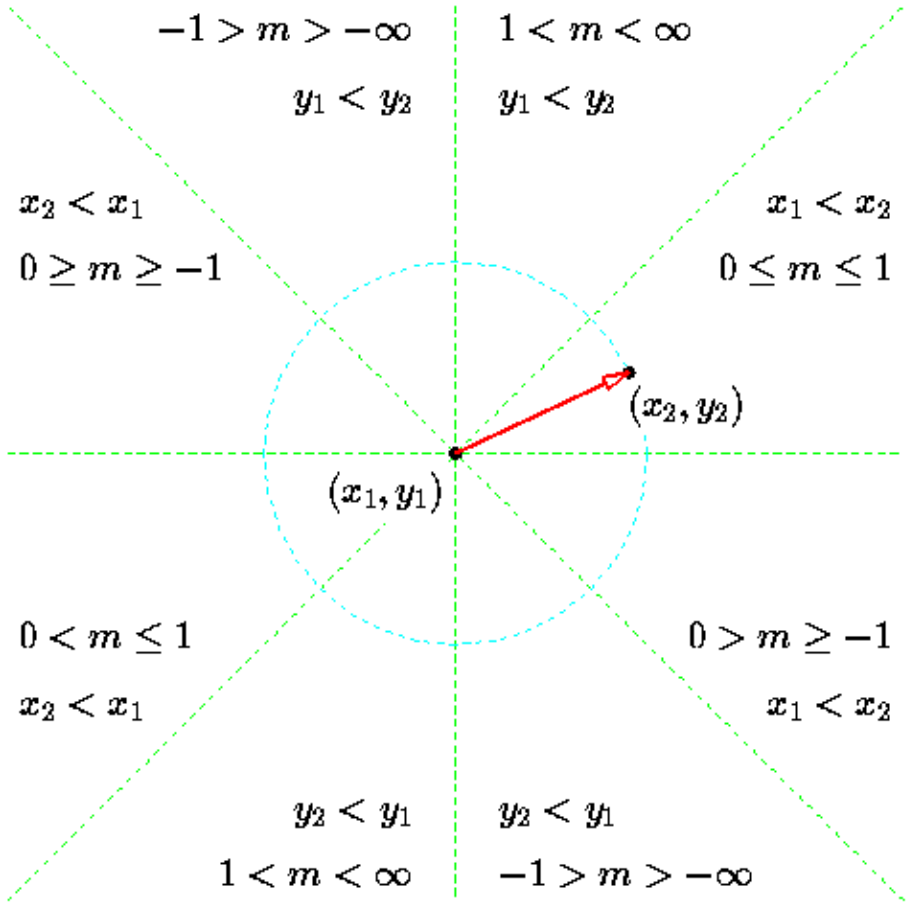
- x=7.1426+(0.8571)=7.9997≈8, y=8+1=9



DDA Problems

- Extremely simple calculations
- Final point is (7.9997,9) instead of (8,9)
- Floating point error

Bresenham's line drawing Algorithm (assumption)



Bresenham's line drawing algorithm (assumption)

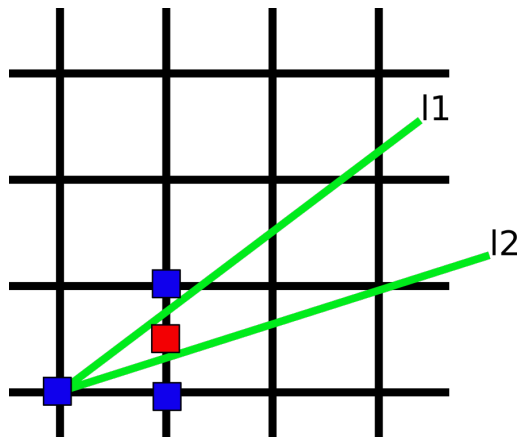
- Two assumptions:
  - $0 < m < 1$
  - $x0 < x1$
- Width W = x2-x1
- Height H = y2-y1

Bresenham's line drawing algorithm (idea)

- W > H (by assumption)
- Increment x by 1
- Two possibilities for y
  - y can stay same
  - Increment y by 1
- Who decides??? - Midpoint algorithm

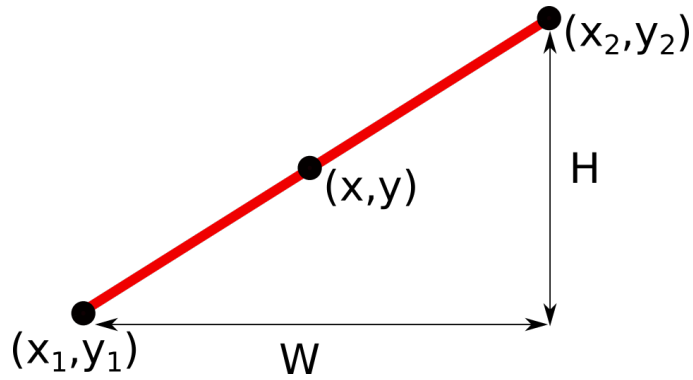
Midpoint algorithm - idea

- W $>$ H (by assumption)
- Increment x by 1
- Two possibilities for y
  - y can stay same $(x_k + 1, y_k)$
  - Increment y by 1 $(x_k + 1, y_k + 1)$
- Who decides??? - Midpoint algorithm



- Midpoint $- >$ $(x_k + 1, y_k + \frac{1}{2})$

Building the algorithm



- We know $\frac{y-y_1}{x-x_1} = \frac{H}{W}$
- $H.(x - x_1) = W.(y - y_1)$
- $H.(x - x_1)$ - $W.(y - y_1) = 0$
- Just multiply by 2 (for avoiding floating point), and call it as F(x,y)
- F(x,y) = $2.H.(x - x_1)$ - $2.W.(y - y_1)$

Building the algorithm

- F(x,y)=2.H.($x - x_1$) - 2.W.($y - y_1$)
- Three possibilities:
  - F(x,y) = 0 - on the line
  - F(x,y) $<$ 0 - above the line
  - F(x,y) $>$ 0 - below the line
- Compute F(x+1,y+$\frac{1}{2}$) and decide accordingly
  - F(x+1,y+$\frac{1}{2}$) $< 0 -> (x_k + 1, y_k + 1)$
  - F(x+1,y+$\frac{1}{2}$) $> 0 -> (x_k + 1, y_k)$

Building the algorithm

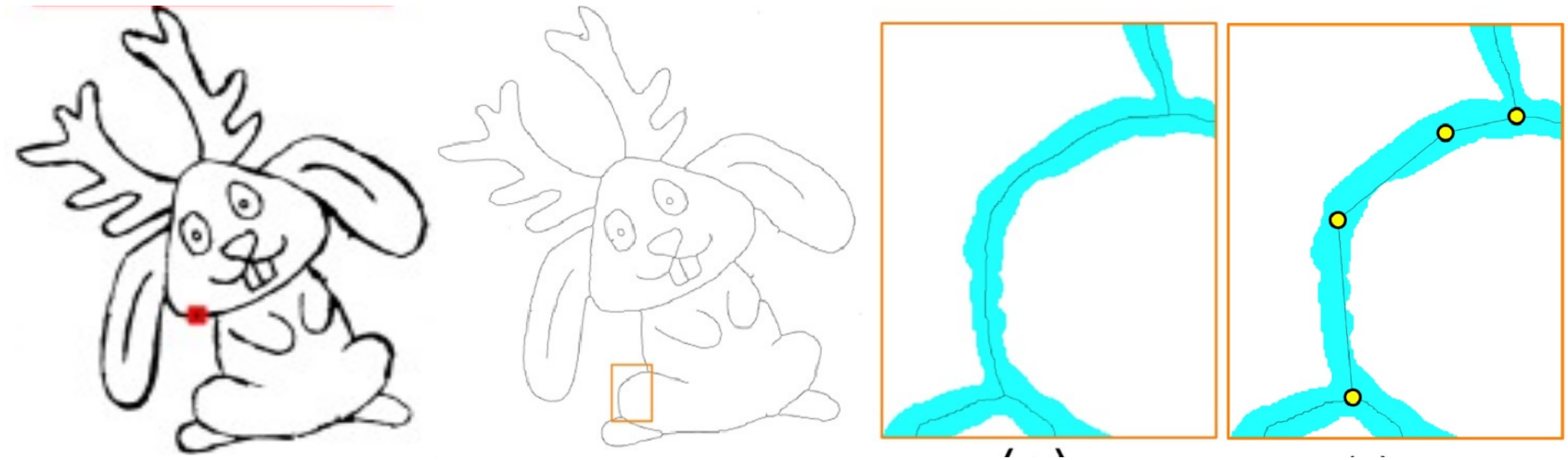
- Initial decision parameter p_0 = F($x_1 + 1, y_1 + \frac{1}{2}$) = 2H(($x_1 + 1$-$x_1$)) - 2W(($y_1 + \frac{1}{2} - y_1$)) = 2H - W
- If we increment to $(x_k + 1, y_k)$, F($x_k + 1, y_k$) = F(x_k, y_k) + 2H
- If we increment to $(x_k + 1, y_k + 1)$, F($x_k + 1, y_k + 1$) = F(x_k, y_k) + 2(H-W)

Bresenham's line drawing Algorithm

Algorithm 2: Bresenham's line drawing Algorithm

Compute H, W, 2H, 2(H-W)
Compute initial decision parameter $F = 2H - W$
Set y as y_1, x as x_1 and initialize k to 0
while $x < x_2$ **do**
    SetPixel(x,y)
    x=x+1
    if F<0 **then**
        F=F+2H
    else
        y=y+1
        F=F+2(H-W)
    end
end

Where to use??



Introduction to OpenGL

Introduction to OpenGL

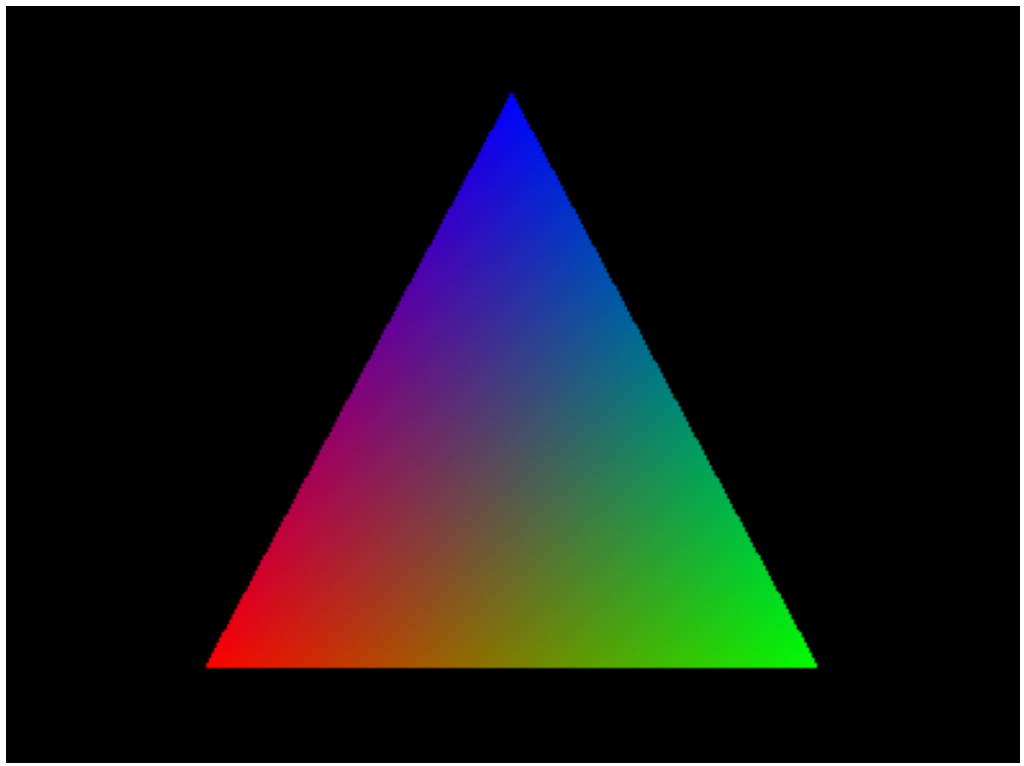
```c
#include <GL/glut.h>

void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
  glColor3f(1, 0, 0); glVertex2f(-0.6, -0.75);
  glColor3f(0, 1, 0); glVertex2f(0.6, -0.75);
  glColor3f(0, 0, 1); glVertex2f(0, 0.75);
  glEnd();
  glFlush();
}

int main(int argc, char** argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(80, 80);
  glutInitWindowSize(400, 300);
  glutCreateWindow("Triangle");
  glutDisplayFunc(display);
  glutMainLoop();
}
```

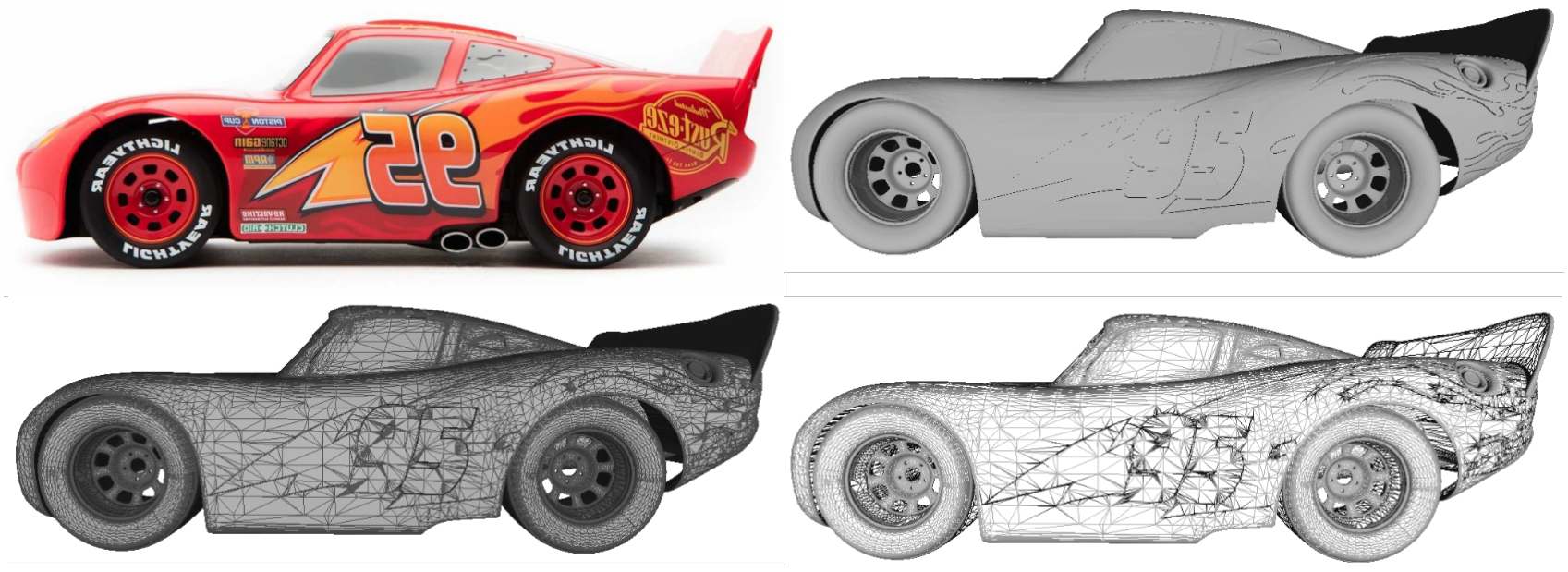
Compiling OpenGL

- Suggestion - Use Linux
- Pre-installed or install freeglut
- Compile using command: gcc filename.cpp -lGL -lGLU -lglut



- OpenGL basic commands

Extra



3D file format

- STL, OBJ, PLY, ...
- STL file sample:

```
solid name
facet normal ni nj nk
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
endsolid name
```

Lab Assignment - I

- Read and display a 3D mesh using C++ and OpenGL!!!
- Deadline: 13^{th} midnight
- @TA's: Please prepare for moodle submission

- Next class:
  - Topic: Circle and ellipse drawing
  - Time: 7^{th} September, 9-10
- Notifications:
  - Roll number 39, 26, 46, 80 $->$ Seminar topic selection deadline is tomorrow midnight (send an email with 3 potential paper titles)
  - Teekaram Meena and Prabhat Kumar want slot change for Monday 10-11 class??
  - TA Change - Sandipan Sharma will be replaced by Ashish Kumar