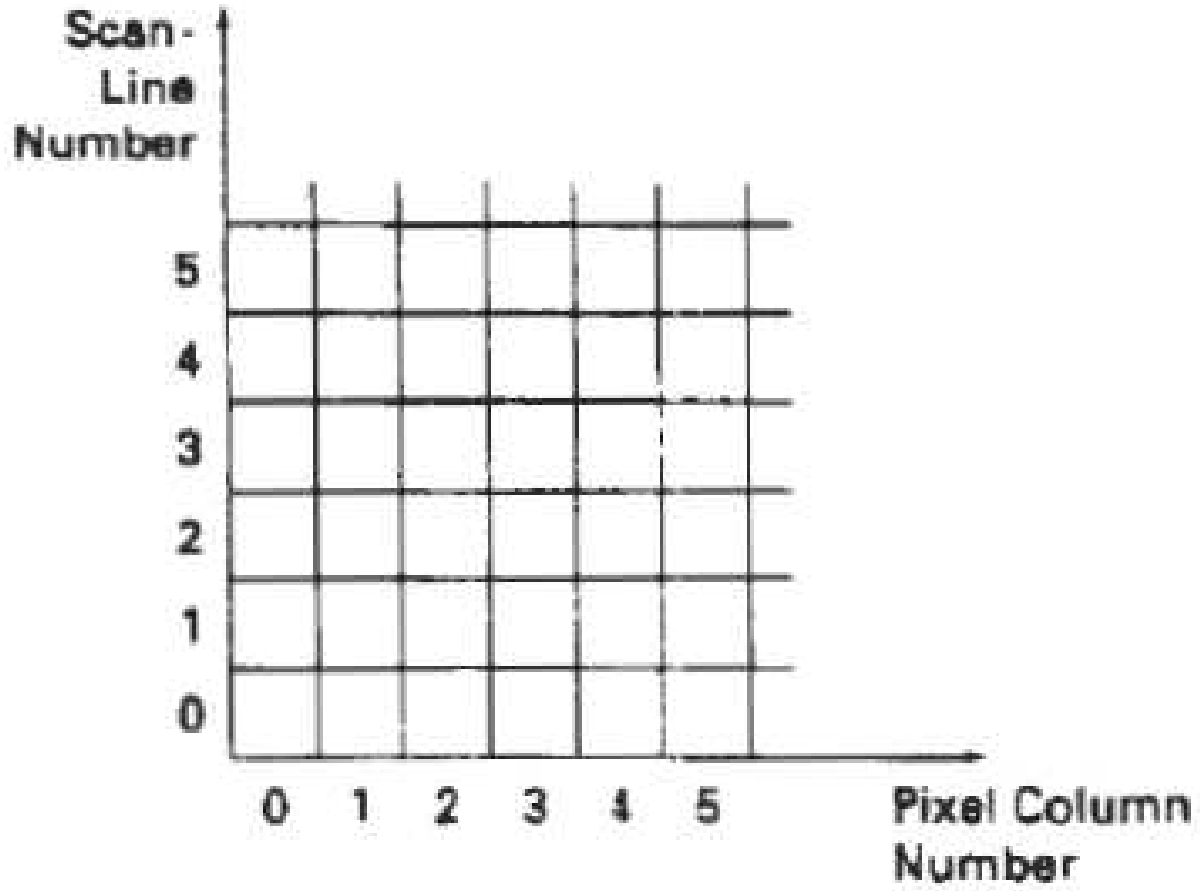
Algoritmos de generación de líneas rectas

Ing Jorge Luis Guevara Díaz
Escuela de Ingeniría de Sistemas
Universidad Privada del Norte

Sistemas de Coordenadas



Algoritmos de generación de líneas

Algoritmo DDA

- Ecuación punto pendiente de la línea

$$y = m \cdot x + b$$

se puede establecer las siguientes relaciones

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - m \cdot x_1$$

Algoritmo DDA

- Si $m = \Delta y / \Delta x$

  Podemos establecer la mínima variación como sigue:

$$m = \delta y / \delta x$$

$$\delta x = \delta y / m$$

$$\delta y = \delta x \cdot m$$

Algoritmo DDA

Si m = Δy/ Δx, entonces

Si m <1

$\delta x = 1$

$y_{k+1}=y_k+ \delta y$

Si m >1

$\delta y = 1$

$x_{k+1}=x_k+ \delta x$

Algoritmo DDA

Si m = -Δy/ Δx, entonces

Si |m| <1

$\delta x = 1$

$y_{k+1}=y_k- \delta y$

Si |m| >1

$\delta y = -1$

$x_{k+1}=x_k+ \delta x$

Algoritmo DDA

Si m = Δy/ -Δx, entonces

Si |m| <1

$\delta x = -1$

$y_{k+1}=y_k+ \delta y$

Si |m| >1

$\delta y = 1$

$x_{k+1}=x_k- \delta x$

Algoritmo DDA

Si m = -Δy/ -Δx, entonces

Si |m| <1

$\delta x = -1$

$y_{k+1}=y_k- \delta y$

Si |m| >1

$\delta y = -1$

$x_{k+1}=x_k- \delta x$

Algoritmo DDA

- Ejercicio dibujar la línea con el algoritmo DDA para los puntos: (x_0, y_o) , (x_{fin}, y_{fin})
  - (0,0)(10,1)
  - (0,0)(1,10)
  - (10,1)(0,3)
  - (10,1)(1,10)
  - (10,1)(0,0)
  - (10,20)(0,0)
  - (1,10)(7,3)
  - (1,3)(7,2)
  - (0,0)(10,10)

laboratorio 2

- laboratorio 1

  programar el algoritmo DDA, en c++ y OpenGL

Programa en openGL

- Funciones

```
void setPixel(GLint xCoord,GLint yCoord)
{
    glBegin(GL_POINTS);
      glVertex2i(xCoord,yCoord);
    glEnd();
}
```

```
void iniciar(void)
{
    glClearColor(1.0,1.0,1.0,0.0);
   //establece color de fondo
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0,200.0,0.0,150.0);

}
```

```
void graficar()
{    glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(1.0,0.0,0.0);

     DDA(0,0,15,125);
     DDA(0,125,15,0);
     DDA(125,0,0,15);
     DDA(125,0,0,300);

     glFlush();
}
```

```
inline int round (const float a){return int(a+0.5);}

void DDA(int x_0,int y_0, int x_fin, int y_fin)
{
    punto2D pixel;

    //analizar si la pendiente m es mayor o menor que 1
    int dx=x_fin-x_0;
    int dy=y_fin-y_0;

    float incremento_y;
    float incremento_x;

    int tam;

    float x=x_0;
    float y=y_0;

    float m=(float)dy/(float)dx;


}
```

```
if(fabs(dy)>fabs(dx))        //pendiente m mayor que 1
    {
        tam=abs(dy);
    }
    else            //pendiente menor que 1
    {
        tam=abs(dx);
    }

    incremento_x=(float)dx/(float)tam;
    incremento_y=(float)dy/(float)tam;

    pixel.setPuntos(x_0,y_0);
    setPixel(pixel.getx(),pixel.gety());

    for(int i=0;i<tam;i++)
    {
     x=x+incremento_x;
     y=y+incremento_y;

     setPixel(round(x),round(y));
    }
```

Algoritmo de Bresenham

Clase punto2D

```
class punto2D
{
    private:
        GLint x,y;
    public:
        punto2D()
        {
            x=y=0;
        }
```

```
void setPuntos(GLint xCoord,GLint yCoord)
    {
        x=xCoord;
        y=yCoord;
    }
    GLint getx() const
    {
         return x;
    }
```

```
GLint gety() const
    {
        return y;
    }
    void incrementarx()
    {
        x++;
    }
    void incrementary()
    {
        y++;
    }

};
```

```
int main(int argc, char** argv)
{

    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowPosition(50,100);

    glutInitWindowSize(400,300);

    glutCreateWindow("Linea");

    iniciar();
    glutDisplayFunc(graficar);
    glutMainLoop();
  return 0;
}
```

Algoritmo de Bresenham

- Falta actualizar esta clase, por favor bajar las diapositivas nuevamente en unos dias