

Laboratorio de Computación Grafica

Pertenece: Erika Judith Tamo Turpo

1.-Dibujo de lineas

Se puede dibujar lineas con puntos. Se considera las variables x_a y y_a puntos iniciales, cuando pulsa el click derecho del mouse y los puntos x,y es cuando se deja de pulsar el click derecho del mouse. Se considero los siguientes métodos:

- Algoritmo Incremental: Es diferente dibujar un linea con pendiente positiva o negativa, para lograrlo se cambia el punto inicial con otro punto o al revez .

```
void linea()  
{  
    glBegin(GL_POINTS);  
    glColor3f(1,0,0);  
    float dy,dx,m;  
    float xd,yd,x1,y1,x2,y2;  
  
    if(y_a>=y and x>=x_a){  
        x2=x;  
        y2=y;  
        x1=x_a;  
        y1=y_a;  
    }  
    if(y_a>=y and x<=x_a){  
        x2=x_a;  
        y2=y_a;  
        x1=x;  
        y1=y;  
    }  
    if(y_a<=y and x_a<=x){  
        x1=x_a;  
        y1=y_a;  
        x2=x;  
        y2=y;  
    }  
}
```

```
dy=y2-y1;  
dx=x2-x1;  
m=dy/dx;  
for(xd=x1;xd<=x2;xd++){  
    yd=(y1+m*(xd-x1));  
    glVertex2f(xd,yd);  
}  
glEnd();  
glFlush();  
}
```

- Linea de DDA

```
void lineaDDA()  
{  
    int xEnd=x;  
    int yEnd=y;  
    int x0=x_a;  
    int y0=y_a;  
    int dx = xEnd - x0, dy = yEnd - y0, steps, k;  
    float xIncrement, yIncrement, x = x0, y = y0;  
    if (fabs (dx) > fabs (dy))//pendiente positiva  
        steps = fabs (dx);  
    else  
        steps = fabs (dy);//pendiente negativa  
    xIncrement = float (dx) / float (steps);  
    yIncrement = float (dy) / float (steps);  
    int r,g,b;  
    r=0;g=0;b=1;  
    glBegin(GL_POINTS);  
    glColor3f(r,g,b);  
    glVertex2f(x,y);  
    glEnd();  
    for ( k = 0; k < steps; k++) {  
        x += xIncrement;  
        y += yIncrement;  
        glBegin(GL_POINTS);  
        glColor3f(r,g,b);  
        glVertex2f(x,y);  
        glEnd();  
    }  
}
```

- Línea de punto medio: Solo se puede dibujar líneas con pendiente entre 0 y 1.

```
void lineapuntomedio()
{
    float dx,dy,x1,y1,x2,y2,d,incE,incNE;
    glBegin(GL_POINTS);
    glColor3f(0,0,1);
    if(x_a<x){
        x2=x;
        y2=y;
        x1=x_a;
        y1=y_a;
    }
    dx=x2-x1;
    dy=y2-y1;
    d=2*dy-dx;
    incE=2*dy;
    incNE=2*(dy-dx);
    glVertex2f(x1,y1);
    while(x1<x2){
        if(d<=0){
            d=d+incE;
            x1=x1+1;
        }
        else{
            d=d+incNE;
            x1=x1+1;
            y1=y1+1;
        }
        glVertex2f(x1,y1);
    }
}
```

Se obtuvo mejor resultado en el DDA.

DDA Line Drawing



2.-Dibujo de Polígono

Es la unión de líneas, primero se pulsa el click del mouse del lado derecho y después se hace una anticlick derecho y dibuja el polígono.

Se guarda los puntos en la matriz “datos del código”.

Traslación

```
if (key==GLUT_KEY_UP){//flecha arriba del teclado,traslación arriba
    up = 1;
    int tamano=datos.size();
    for(int i=0;i<tamano;i++)
    {
        datos[i][1]+=desplazamiento;
    }
    plot_polygon();
}
if (key==GLUT_KEY_DOWN){//flecha arriba del teclado,traslación abajo
    down = 1;
    int tamano=datos.size();
    for(int i=0;i<tamano;i++)
    {
        datos[i][1]-=desplazamiento;
    }
    plot_polygon();
}
if (key==GLUT_KEY_LEFT){//flecha arriba del teclado,traslación izquierda
    down = 1;
    int tamano=datos.size();
    for(int i=0;i<tamano;i++)
    {
        datos[i][0]-=desplazamiento;
    }
    plot_polygon();
}
```

Escalamiento

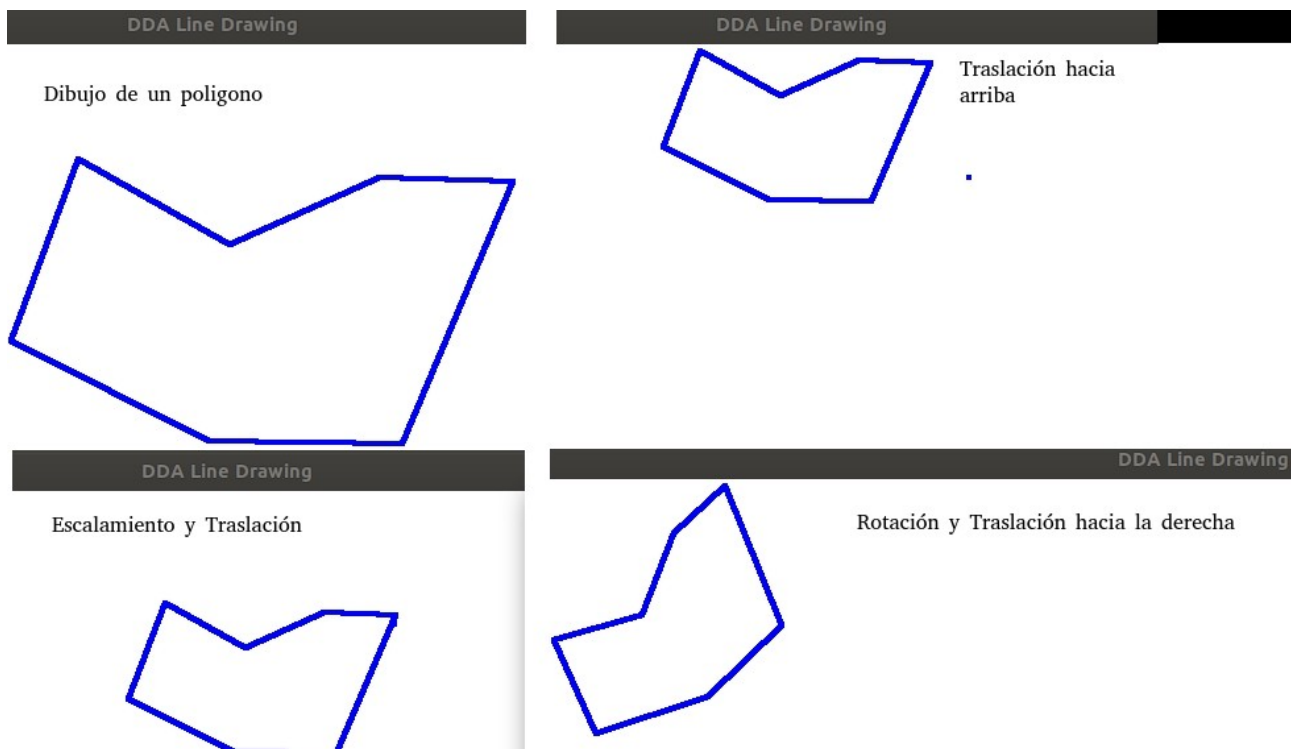
```
if(key==115){//tecla control(ctrl), escalamiento negativo
    int tamano=datos.size();
    for(int i=0;i<tamano;i++)
    {
        datos[i][0]*= escala_me;
        datos[i][1]*= escala_me;
    }
    plot_polygon();
}

if(key==113){//tecla shift(flecha para arriba), escalamiento positivo
    cout<<"imprimiendo más deberas en enl maas"<<endl;
    int tamano=datos.size();
    for(int i=0;i<tamano;i++)
    {
        datos[i][0]*= escala_gra;
        datos[i][1]*= escala_gra;
    }
    plot_polygon();
    for(int i=0;i<tamano;i++)
    {
        cout<<datos[i][0]<<endl;
        cout<<datos[i][1]<<endl;
    }
}
```

Rotación

```
if(key==116){
    int tamaño=datos.size();
    for(int i=0;i<tamaño;i++)
    {
        int next=i+1;
        cout<<"x="<<datos[i][0]<<endl;
        cout<<"y="<<datos[i][1]<<endl;
        float temp=datos[i][0];
        datos[i][0]=200+(datos[i][0]-200)*sqrt(2)/2 - (datos[i][1]-200)*sqrt(2)/2;
        datos[i][1]=200+(temp-200)*sqrt(2)/2 + (datos[i][1]-200)*sqrt(2)/2;
        cout<<" " <<datos[i][0]<<endl;
        cout<<" " <<datos[i][1]<<endl;
    }
    plot_polygon();
}
```

Resultado



Rellenado de polígonos

Existen algunas limitaciones por ejemplo se tiene que crear dos listas lo suficientemente grandes obtener posiciones de píxeles a pintar.

```

void PintarPoligono::detectarBorde(float x1,float y1,float x2,float y2,int *le,int *re){
    float temp,x,mx;
    int i;
    if(y1>y2){
        temp=x1,x1=x2,x2=temp;
        temp=y1,y1=y2,y2=temp;
    }
    if(y1==y2)
        mx=x2-x1;
    else
        mx=(x2-x1)/(y2-y1);
    x=x1;
    for(i=int(y1);i<=(int)y2;i++){
        if(x<(float)le[i]) le[i]=(int)x;
        if(x>(float)re[i]) re[i]=(int)x;
        x+=mx;
    }
}

```

```

void PintarPoligono::rellenar(vector<vector<float>> V){
    int le[500],re[500],i,j;
    for(i=0;i<500;i++){
        le[i]=500,re[i]=0;
    }
    int p=0;
    for(p=0;p<V.size()-2;p++){
        detectarBorde(V[p][0],V[p][1],V[p+1][0],V[p+1][1],le,re);
    }
    detectarBorde(V[p][0],V[p][1],V[0][0],V[0][1],le,re);
    for(j=0;j<500;j++){
        {
            if(le[j]<=re[j])
                for(i=le[j];i<re[j];i++)
                    draw_pixel(i,j);
        }
    }
}

```

Obteniendo el resultado siguiente:



3.-Dibujo de Circulos

En este caso cuando se hace click derecho del mouse se obtienen las coordenadas de ese punto nombradas como x y y.

Se utilizó el algoritmo de punto medio que dibuja un $\frac{1}{4}$ del círculo del primer cuadrante:

```

void circleMidpoint2(float x, float y, int r){
    float d, temp_x, temp_y;
    temp_x=0;
    temp_y=r;
    d=5/4-r;
    int ro,g,b;
    ro=1;g=0;
    b=0;
    circlePlotPoints(x,y,temp_x,temp_y);
    while (temp_y>temp_x){
        if(d<0){
            d=d+2*temp_x+3;
            temp_x++;
        }
        else{
            d=d+2*(temp_x-temp_y)+5;
            temp_x++;
            temp_y--;
        }
        circlePlotPoints(x,y,temp_x,temp_y);
    }
    glEnd ( );
}

```

Para poder dibujar completo el círculo se hace lo siguiente:

```

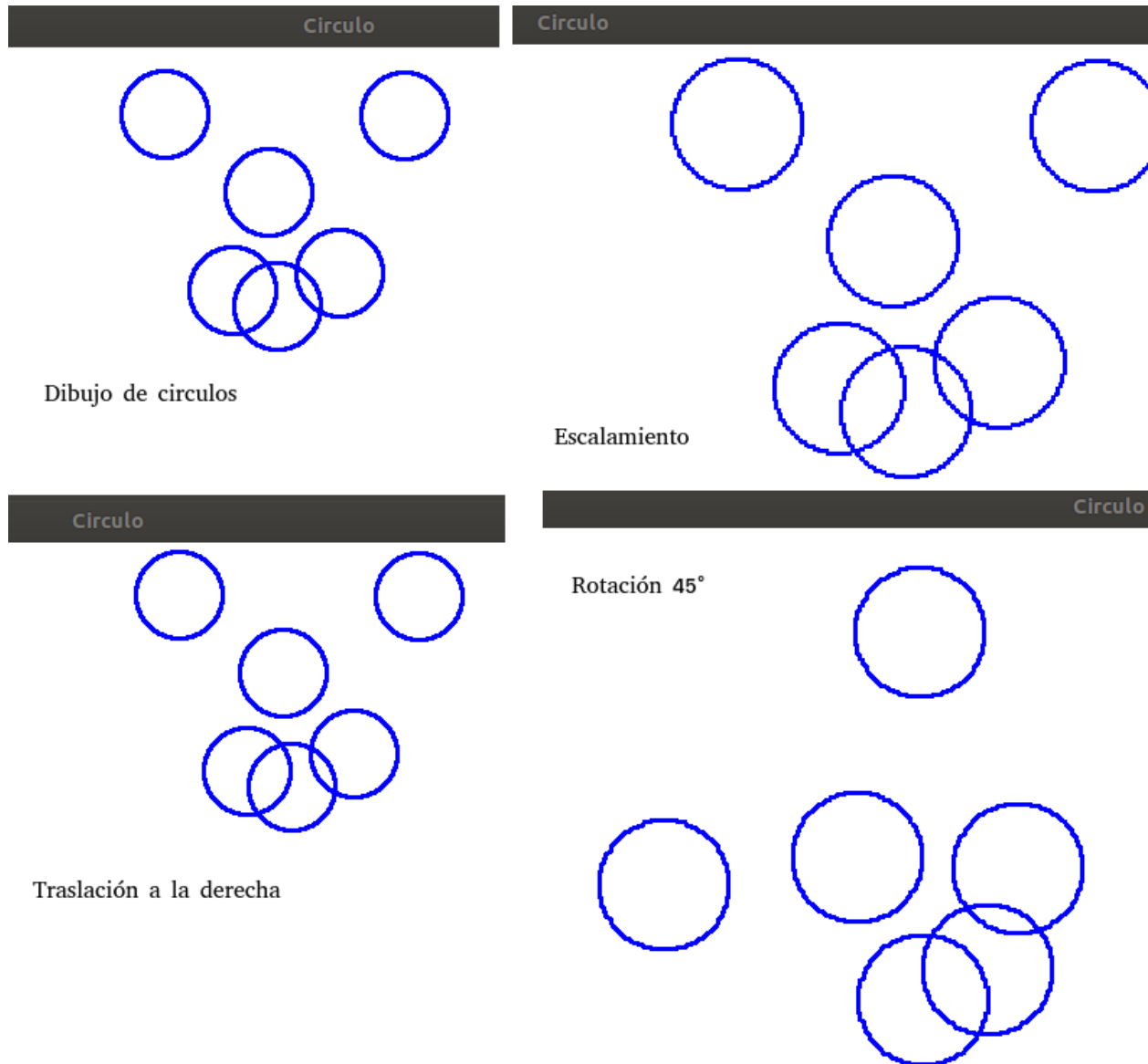
void setPixel (float xCoord, float yCoord)
{
    glBegin (GL_POINTS);

    vector<float> punto;
    punto.push_back(xCoord);
    punto.push_back(yCoord);
    datos.push_back(punto);
    glColor3f(ro,g,b);
    glVertex2f (xCoord, yCoord);
    glEnd ( );
}

void circlePlotPoints (float x, float y, float tx, float ty)
{
    setPixel (x + tx, y + ty);
    setPixel (x - tx, y + ty);
    setPixel (x + tx, y - ty);
    setPixel (x - tx, y - ty);
    setPixel (x + ty, y + tx);
    setPixel (x - ty, y + tx);
    setPixel (x + ty, y - tx);
    setPixel (x - ty, y - tx);
    glutPostRedisplay();
}

```

Resultado



4.- Línea 3D

Se logro programar con la técnica incremental.

```
void linea(float x1, float y1, float z1, float x2, float y2, float z2)
{
    glBegin(GL_POINTS);
    float m, x, y, z, dy, dx;
    dy=y2-y1;
    dx=x2-x1;
    m=dy/dx;
    for(x=x1; x<=x2; x+=0.01){
        y=(y1+m*(x-x1));
        glVertex3f(x, y, x);
    }
    glEnd();
    glFlush();
}
```

Obteniendo el siguiente resultado:

