Universidad Nacional de San Antonio Abad del Cusco
Departamento Académico de Informática
COMPUTACIÓN GRAFICA I
Práctica Nº 3

ALGORITMO DE RECORTE DE LINEAS DE COHEN-SUTHERLAND

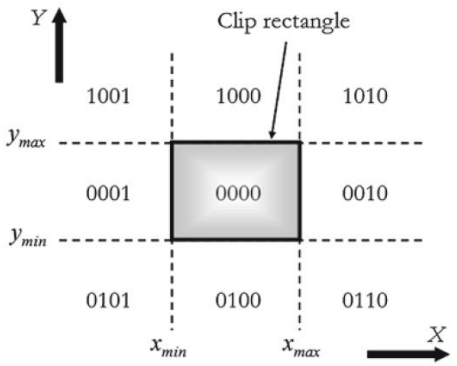
1. **OBJETIVO.**
   - Conocer el algoritmo de recorte de líneas de Cohen-Sutherland.
   - Implementar el algoritmo de Cohen-Sutherland en OpenGL.

2. **ALGORITMO DE COHEN-SUTHERLAND**

Este algoritmo realiza varias comprobaciones iniciales para descubrir si se puede evitar cálculos de las intersecciones. El primer paso es comprobar si los puntos extremos del segmento son aceptados por sus posiciones. Si esto no es posible, entonces realizamos varias comprobaciones por regiones exteriores al rectángulo de recorte, formadas por las líneas rectas al extender sus aristas. Asimismo, podemos rechazar un segmento inmediatamente si sus extremos yacen en regiones a la izquierda de x_{min}, por encima de y_{max}, a la derecha de x_{max}, y por debajo de y_{max}.

Si el segmento no se puede aceptar ni rechazar inmediatamente, entonces es partido en dos segmentos por una arista del rectángulo de recorte, para que uno de ellos sea rechazado de inmediato. Esto implica que implementamos un método iterativo de recorte para el segmento hasta que éste sea aceptado o rechazado. Este algoritmo es eficiente especialmente en el caso de tener un rectángulo de recorte muy grande que comprende a todos o una gran mayoría de los segmentos a visualizar. Asimismo, podemos tener el caso de un rectángulo de recorte muy pequeño que fuerza a todos o casi todos los segmentos a ser rechazados.

Se le asigna un código de cuatro bits, $b_3b_2b_1b_0$, a cada región de las nueve creadas. Este código binario es creado estratégicamente para representar rápida y fácilmente cada región. El valor de 1 indicará que un extremo yace en tal región mientras que un 0 indicará lo contrario:



3. DESARROLLO DE LA PRÁCTICA.

3.1. EJERCICIO 1.

Compilar y ejecutar el siguiente programa para hacer recorte de líneas según el algoritmo de Cohen-Sutherland.

```
#include<math.h>
#include<cstdlib>
#include<GL/glew.h>
#include<GL/glut.h>
#include<stdio.h>
#include<iostream>

//--cohen sutherland

#define MAX 20
//--outcodes de los lados del rectángulo en hexadecimal
enum { TOP = 0x1, BOTTOM = 0x2, RIGHT = 0x4, LEFT = 0x8 };
int ln[MAX][4]; //--puntos extremos de las líneas
int xmin, ymin, xmax, ymax; //--coordenadas del rectángulo de recorte
int lnclip[MAX][4];  //--lineas recortadas
int n; //--nro de lineas
int k;
typedef unsigned int outcode;

using namespace std;
void Plot(int ix, int iy)
{
	glBegin(GL_POINTS);
	glVertex2i(ix, iy);
	glEnd();
}
void swap(int* x, int* y)
{
	int aux = *x;
	*x = *y;
	*y = aux;
}

void dibujaLinea(int x0, int y0, int x1, int y1) //--bresenham
{
	int dy, x, y, error;
	int delta_x, delta_y;
	bool steep = abs(y1 - y0) > abs(x1 - x0);
	if (steep == true)
	{
		swap(&x0, &y0);
		swap(&x1, &y1);
	}
	if (x0 > x1)
	{
		swap(&x0, &x1);
		swap(&y0, &y1);
	}
	if (y0 > y1)
		dy = -1;
	else
		dy = 1;

	delta_x = x1 - x0;
	delta_y = abs(y1 - y0);
	y = y0;
	error = 0;
	for (x = x0; x <= x1; x++)
	{
		if (steep == true)
			Plot(y, x);
		else
```

```
                Plot(x, y);
            error = error + delta_y;
            if (2 * error >= delta_x)
            {
                y = y + dy;
                error = error - delta_x;
            }
        }

}
void dibujaRectangulo(float xmin, float ymin, float xmax, float ymax)
{  //--dibuja un rectángulo desde xmin, ymin hasta xmax, ymax
        //LEFT
        dibujaLinea(xmin, ymin, xmin, ymax);
        //TOP
        dibujaLinea(xmin, ymax, xmax, ymax);
        //RIGHT
        dibujaLinea(xmax, ymax, xmax, ymin);
        //BOTTOM
        dibujaLinea(xmax, ymin, xmin, ymin);
}

outcode calculo_outcode(int x, int y, int xmin, int ymin, int xmax, int ymax)
{ //--calcula el outcode de un punto x,y con referencia la rectángulo de
  //--recorte xmin, ymin, xmax, ymax

        outcode oc = 0;

        if (y > ymax)
                oc = oc | TOP;
        else if (y < ymin)
                oc = oc | BOTTOM;

        if (x > xmax)
                oc = oc | RIGHT;
        else if (x < xmin)
                oc = oc | LEFT;
        return oc;
}

void cohen_sutherland(double x0, double y0, double x1, double y1,
        double xmin, double ymin, double xmax, double ymax)
{   //--determina los recortes de lineas
        int in;
        int done;
        outcode outcode0, outcode1;

        in = FALSE;
        done = FALSE;

        outcode0 = calculo_outcode(x0, y0, xmin, ymin, xmax, ymax);
        outcode1 = calculo_outcode(x1, y1, xmin, ymin, xmax, ymax);
        while (done == FALSE)
        {
                if ((outcode0 | outcode1) == 0)
                {
                        done = TRUE;
                        in = TRUE;
                }
                else if ((outcode0 & outcode1) != 0)
                {
                        done = TRUE;
                }
                else
                {
                        double m = (y1 - y0) / (x1 - x0);
                        double x, y;

                        int outcode;
```

```
                        if (outcode0 != 0)
                                outcode = outcode0;
                        else
                                outcode = outcode1;

                        if ((outcode0 & TOP) != 0)
                        {
                                x = x0 + (ymax - y0) / m;
                                y = ymax;
                        }

                        else if ((outcode & BOTTOM) != 0)
                        {
                                x = x0 + (ymin - y0) / m;
                                y = ymin;
                        }
                        else if ((outcode & RIGHT) != 0)
                        {
                                y = y0 + m * (xmax - x0);
                                x = xmax;
                        }
                        else
                        {
                                y = y0 + m * (xmin - x0);
                                x = xmin;
                        }
                        if (outcode == outcode0)
                        {
                                x0 = x;
                                y0 = y;
                                outcode0 = calculo_outcode(x0, y0, xmin, ymin, xmax, ymax);
                        }
                        else
                        {
                                x1 = x;
                                y1 = y;
                                outcode1 = calculo_outcode(x1, y1, xmin, ymin, xmax, ymax);
                        }
                }
        }//-- fin while;

        //--lineas despues de clipping
        cout << x0 << "," << y0 << "---" << x1 << "," << y1 << endl;
        if (in == TRUE) //--la linea es aceptada
        {//--guardamos sus coordenadas en una matriz
                lnclip[k][0] = x0;
                lnclip[k][1] = y0;
                lnclip[k][2] = x1;
                lnclip[k][3] = y1;
                k += 1;
        }
}
//--dibuja en la segunda ventana el recorte de líneas
void display2()
{
        //--dibuja el rectangulo de recorte después de recorte
        glClear(GL_COLOR_BUFFER_BIT);
        dibujaRectangulo(xmin,ymin,xmax,ymax);

        for (int i = 0; i < k; i++)
        {
           dibujaLinea(lnclip[i][0], lnclip[i][1], lnclip[i][2], lnclip[i][3]);
        }
        glFlush(); //--fuerza la ejecución de los comandos de OpenGL
}

//--dibuja en la primera ventana las linas antes del recorte
void display1()
{
```

```
	glClear(GL_COLOR_BUFFER_BIT);
	//--dibujar rectángulo de recorte y líneas antes de clipping
	dibujaRectangulo(xmin,ymin,xmax,ymax);
	for (int i = 0; i < n; i++)
	{
		dibujaLinea(ln[i][0], ln[i][1], ln[i][2], ln[i][3]);
	}

	glFlush(); //--fuerza la ejecución de los comandos de OpenGL
}
void myinit()
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glColor3f(1.0, 0.0, 0.0);
	glPointSize(1.0); //--tamaño de los puntos
	glMatrixMode(GL_PROJECTION); //--activar la matriz de transformación
	glLoadIdentity(); //--cargar la identidad y poder hacer las transformaciones
	gluOrtho2D(0.0, 499.0, 0.0, 499.0); //--establece una ventana de dibujo 2D
}

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

	int i, j;
	k = 0;
	cout << "Ingrese el número de lineas a recortar: "<<endl;
	cin >> n;

	cout << "Ingrese las coordenadas x - y de los puntos extremos de las lineas:\n";
	for (i = 0; i < n; i++)
	{
		cout << "INGRESE PUNTOS DE LA LINEA " << i+1 <<":"<< endl;
		for (j = 0; j < 4; j++)
		{
			if (j % 2 == 0)
				cout << "Coordenada X:";
			else
				cout << "Coordenada Y:";
			cin >> ln[i][j];
		}
	}

	cout << "INGRESE LAS COORDENADAS X-Y DEL RECTANGULO DE RECORTE:"<<endl;
	cout << "Ingrese Xmin:";
	cin >> xmin;
	cout << "Ingrese Ymin";
	cin >> ymin;
	cout << "Ingrese Xmax:";
	cin >> xmax;
	cout << "Ingrese Ymax";
	cin >> ymax;
	 //--recorta las líneas
	for (i = 0; i < n; i++)
	{
	   cohen_sutherland(ln[i][0], ln[i][1], ln[i][2], ln[i][3], xmin, ymin, xmax, ymax);
	}
	/*  Inicializacion GLUT estándar*/
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	/*definición de ventana 1*/
	glutInitWindowSize(500, 500); /* ventana 500x500 pixeles */
	glutInitWindowPosition(0, 0); /* coloca la ventana de despliegue en esq. sup. izq */
	glutCreateWindow("LINEAS ANTES DEL RECORTE"); /* título de la ventana*/
	glutDisplayFunc(display1); /*llama a la funcion display cuando se abre la ventana */
	myinit(); /* fija o establece los atributos */

	/*definición de ventana 2*/
	glutInitWindowSize(500, 500); /* ventana 500x500 pixeles */
	glutInitWindowPosition(550, 0);
```

```
        glutCreateWindow("LINEAS DESPUES DEL RECORTE"); /* título de la ventana*/
        glutDisplayFunc(display2); /*llama a la función display cuando se abre la ventana */
        myinit(); /* fija o establece los atributos */
        glutMainLoop(); /* entra a un ciclo de evento */
}
```

4. TAREA.

4.1. Implementar en OpenGL el algoritmo parámetrico estudiado en clases para realizar recorte de líneas.

5. BIBLIOGRAFIA

5.1. Donald Hearn, M. Pauline Baker; Gráficas por Computadora con OpenGL; Pearson Prentice Hall; 3° edición; Madrid 2006.

5.2. Rimon Elias; Digital Media A problema-solving Aproach for Computer Graphics; eBook. New York

5.3. Sumanta Guha; Computer Graphics Through OpenGL; Third Edition; Taylor & Francis Group . 2019. NW.