Alumna: Ruth Esther Huillca Roque

Tema: Ejercicios Computación Grafica

Ejercicio 1.6

Escriba los procedimientos inp_to_ndc, ndc_to_user, user_to_ndc y ndc_to_dc, que transforman datos entre los sistemas de coordenadas, como se muestra en la Figura 1.3. Repita el ejercicio asumiendo que el intervalo de variación del sistema NDC va de:

(i) -1 a +1 (coordenadas normalizadas centradas)

inp_to_ndc

```
ndcx = dcx / ndhm1;
ndcy = dcy / ndvm1;
```

ndc_to_user

```
x = ( ndcx * (xmax - xmin) ) + xmin;
y = ( ndcy * (ymax - ymin) ) + ymin;

x = ( ndcx * (1 - (-1)) ) + (-1);
y = ( ndcy * (1 - (-1)) ) + (-1);
```

user_to_ndc

```
ndcx = (x - xmin) / (xmax - xmin);
ndcy = (y - ymin) / (ymax - ymin);

ndcx = (x - (-1)) / (1 - (-1));
ndcy = (y - (-1)) / (1 - (-1));
```

ndc_to_dc

```
dcx = round(ndcx * ndhm1);
dcy = round(ndcy * ndvm1);
```

(ii) 0 a 100

inp_to_ndc

```
                ndcx = dcx / ndhm1;
                ndcy = dcy / ndvm1;
ndc_to_user
                x = ( ndcx * (xmax - xmin) ) + xmin;
                y = ( ndcy * (ymax - ymin) ) + ymin;

                x = ( ndcx * (100 - (0)) ) + (0);
                y = ( ndcy * (100 - (0)) ) + (0);
user_to_ndc
                ndcx = (x - xmin) / (xmax - xmin);
                ndcy = (y - ymin) / (ymax - ymin);

                ndcx = (x - (0)) / (100 - (0));
                ndcy = (y - (0)) / (100 - (0));
        ndc_to_dc
                dcx = round(ndcx * ndhm1);
                dcy = round(ndcy * ndvm1);
```

Ejercicio 2.4

Calcule las razones de aspecto (gráfica y física), y las resoluciones de área horizontal y vertical de una pantalla de TV a color estándar, donde:

- width = 42cm;
- height = 31cm;
- ndh = 546;
- ndv = 434.

1. Escriba un programa (en C) que dibuje un polígono regular (es decir, un polígono con lados de longitudes iguales y ángulos internos iguales) con un radio igual a un tercio de la altura de la

pantalla, y centro en el centro de la pantalla. El programa debe solicitar el número de vértices, n.

```
#include <GL/glut.h>
#include <math.h>
#include <iostream>
#define M_PI 3.14159265358979323846
using namespace std;

int num_lados;
double radio;

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,0,1);
	glLoadIdentity();
	GLfloat grados_1;
	GLfloat grados_2;
	glBegin(GL_LINES);
  //double radio = 0.2;
	double puntos = 360.0f/num_lados;

  for(double i=0;i<360;i=i+puntos){
		grados_1 = (GLfloat)i*M_PI/180.0f;
		grados_2 = ((GLfloat)i+puntos)*M_PI/180.0f;
		glVertex3f(radio*cos(grados_1),radio*sin(grados_1),0.0f);
		glVertex3f(radio*cos(grados_2),radio*sin(grados_2),0.0f);
	}
```

```
        glEnd();
        glFlush();
}

void reshape(int width, int height)
{
        glViewport(0, 0, width, height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1, 1, -1, 1, -1, 1);
        glMatrixMode(GL_MODELVIEW);
}

void init()
{
        glClearColor(0,0,0,0);
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        cout<<"Ingrese el numero de lados"<<endl;
        cin>>num_lados;
   cout<<"Ingrese radio:"<<endl;
        cin>>radio;
        glutInitWindowPosition(50, 50);
        glutInitWindowSize(600, 600);
        glutCreateWindow("Poligono");
```

```
        init();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
        return 0;
}
```

2. Modifique el programa del ejercicio anterior para solicitar también en el radio r del cırculo que circunscribe el polıgono. Fa¸ca con que su programa estime el n nero n de vértices necesarios para que el polıgono pare¸ca una buena aproximación a un c ırculo para diferentes rayos. Utilizando los parámetros ndh, ndv, width, height del dispositivo, obtenga una relación teórica entre n o r para la aproximación al de círculos a través de polıgonos.

Ndh: Es la cantidad de posiciones que pueden ser direccionables de manera horizontal.

Ndv: Es la cantidad de posiciones que pueden ser direccionables de manera vertical.

Width: Ancho del rectángulo en mm.

Height: Altura del rectángulo en mm.

Teniendo en cuenta todo ello, vamos a hallar la relación teórica para la aproximación de círculos utilizando los polígonos:

angulo = 2*PI / n → n es el número de vértices.

Por otro lado hallaremoss el radio mediante cosenos y senos del angulo

radio * cos (angulo)

radio * sin (angulo)

3. Algunos dispositivos vectoriales ofrecen un conjunto de tres primitivas graficas:

• pen up: levanta la pluma del papel, o apaga el haz de los etrons;

• pen down: coloca la pluma sobre el papel, o enciende el haz;

• locate (dcx, dcy): coloca la CP en un punto del rectángulo de visualización.

Escriba rutinas de software que simule estas tres primitivas, usando las primitivas de movimiento y de dibujo.

```
pen
{
        mover_a(dcx,dcy)
        dibujar_a(dcx,dcy)
}
pen_down
{
        dibujar_a(dcx,dcy)
        dibujar_a(dcx,dcy)
 }
buscar(dcx, dcy)
 {
        mover_a(dcx, dcy);
 }
```

4. Calcule las razones de aspecto (gráfica y física), y las resoluciones de área horizontal y vertical de una pantalla

de TV a color estándar, donde:

• width = 42cm;

• height = 31cm;

• ndh = 546;

• ndv = 434.

1. **resolución horizontal:** tenemos que dividir el número de posiciones que pueden ser direccionables de manera horizontal con el ancho del rectángulo:

horiz_res = ndh/width = 546/420

2. **tamaño punto horizontal:** dividimos el ancho del rectángulo con el número de posiciones que pueden ser direccionables de manera horizontal:

   horiz_punto_tam = width/ndh = 420/546

3. **resolución vertical:** dividimos el número de posiciones que pueden ser direccionables de manera vertical con la altura del rectángulo:

   vert_res = ndv/height = 434/310

4. **tamaño punto vertical**: Se divide la altura del rectangulo con el numero de posiciones qu pueden ser direccionables de manera vertical:

   tam_punt_vert = height/ndv = 310/434

5. **total puntos direccionables:** Se multiplica el numero de posiciones que pueden ser direccionables de manera horizontal con el numero de posiciones que pueden ser direccionables de manera vertical:
   total_puntos = ndh.ndv = 546*434
6. **resolución de área:** Divido el numero total de puntos con la multiplicación de la altura y ancho del rectangulo:
   área res = total_puntos/(width.height) = (546*434)/(420*310)
7. **razón de aspecto gráfico:** divido el tamaño del punto vertical con el tamaño del punto horizontal:
   aspect_ratio = tam_punt_vert/horiz _punto_tam = (310/434)/(420/546)
8. **razón de aspecto fısico:** Divido la altura entre ancho del rectangulo
   physical_aspect_ratio = height/width = 310/420