Ejercicios 2.4

Lorena Xiomara Castillo Galdos

Septiembre 2018

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

```
#include <GL/glut.h>
#include <math.h>
#include <cmath>
double height= 300;
double radio = (height/300)*2 /3 ;
using namespace std;
void drawPoly(int numLados)
{
        double  temp_x=radio*cos(2*M_PI/numLados),
                temp_y=radio*sin(2*M_PI/numLados);
        for(int i =0; i <= numLados; i++)
        {
                double angulo = i*2*M_PI/numLados;
                double x = radio*cos(angulo);
                double y = radio*sin(angulo);
                glBegin(GL_LINES);
                glVertex2d(x,y);
                glVertex2d(temp_x,temp_y);
                glEnd();
                temp_x=x;
                temp_y=y;
        }
}
void displayMe(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        drawPoly(100);
                glFlush();
}
int main(int argc, char** argv)
```

```
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE);
        glutInitWindowSize(300, 300);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Ejercicio");
        glutDisplayFunc(displayMe);
        glutMainLoop();
        return 0;
}
```

2. Modifique el programa del ejercicio anterior para pedir también el radio r del círculo que circunscribe el polígono.

```
#include <GL/glut.h>
#include <math.h>
#include <cmath>
using namespace std;
void drawPoly(double radio,int numLados)
{
        double  temp_x=radio*cos(2*M_PI/numLados),
                temp_y=radio*sin(2*M_PI/numLados);
        for(int i =0; i <= numLados; i++)
        {
                double angulo = i*2*M_PI/numLados;
                double x = radio*cos(angulo);
                double y = radio*sin(angulo);
                glBegin(GL_LINES);
                glVertex2d(x,y);
                glVertex2d(temp_x,temp_y);
                glEnd();
                temp_x=x;
                temp_y=y;
        }
}
void displayMe(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        drawPoly(0.5,100);
                glFlush();
}
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE);
        glutInitWindowSize(300, 300);
```

```
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Ejercicio");
        glutDisplayFunc(displayMe);
        glutMainLoop();
        return 0;
}
```

3. Algunos dispositivos vectoriales ofrecen un conjunto de tres primitivas gráficas: pen up (levanta la pluma del papel), pen down (coloca la pluma sobre el papel), locate (posiciona 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_up()
{
    moveto(dcx,dcy)
}
pen_down()
{
    moveto(dcx,dcy)
    drawto(dcx,dcy)
}
locate(dcx, dcy)
{
    moveto(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 = 420 mm;
   - height = 31cm = 310 mm;
   - ndv = 434.
   - ndh = 546;

```
razonAspectoGrafico = (height/ndh) / (width/ndv)
razonAspectoGrafico = (310/546) / (420/434)
razonAspectoFisico = height / width
razonAspectoFisico = 310 / 420
resolucionHorizontal = ndh / width
resolucionHorizontal = 546 / 420
resolucionVertical = ndv / height
resolucionVertical = 434 / 310
```