# PRACTICA Nº 3. VENTANA Y PUERTO DE VISTA

## 3.1 Dibujar la función sinc, utilizando Puerto de Vista en OpenGL

Se trata de representar la función *sinc*, (Ver apartado 2 de apuntes "Ventana y Puerto de Vista"). Con OpenGL lo que se define es la Ventana y el Puerto de Vista. A continuación se muestra el código asumiendo que queremos mostrar la curva para valores de $-4 \leq x \leq 4$ en un puerto de vista de 640 x 480. En la ventana para que se vea más centrado vamos a tomar un rango algo mayor.

```
void myDisplay(void)
{
//Dibuja la función
setWindow(-5, 5, -0.3, 1.1);
setViewport(0,640,0,480);
glBegin(GL_LINE_STRIP);
for(GLdouble x =-4.0; x < 4.0; x += 0.05)
                {
                        GLfloat y =sin(GL_PI*x)/(GL_PI*x);
                        glVertex2f(x,y);
                }
                        glEnd();
```

## 3.2 Dibujar una polilínea a partir de un archivo

En la práctica 1.5 vimos como se representaba una polilínea a partir de un archivo externo, en el caso del ejemplo "*dino.dat*". No fue necesario introducir el concepto de Ventana y Puerto de Vista puesto que los valores del archivo se representaban correctamente para los valores de ventana de 640x480, ya que todos los datos están comprendidos entre (0, 0) y (640, 480). No obstante mediante el Puerto de Vista se puede tener más control sobre la situación.

Se trata en esta prácticas de representar en una misma ventana diferentes Puertos de Vista de la misma imagen posicionados de una forma matricial como muestra la figura 1.

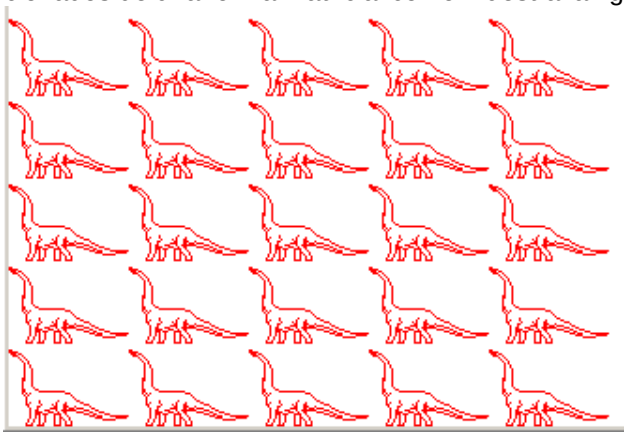

Figura 1 Copias de dinosaurio al posicionar diferentes puertos de vista

```
void myDisplay(void)
{
glClear(GL_COLOR_BUFFER_BIT);
setWindow(0,640.0,0,480.0);
for(int i=0; i< 5; i++)
     for(int j=0;j<5;j++)
       {
         glViewport(i*128,j*96,128,96);
        drawPolyLineFile("dino.dat");
       }
}
```

NOTA: Cada copia se dibuja en un PV de tamaño 128x96 pixels con relación de aspecto 640/480, así no se distorsiona el dinosaurio.

## 3.3 Puerto de vista girado 180º

Realizar las modificaciones necesarias en el código anterior para producir el efecto mostrado en la figura 2.

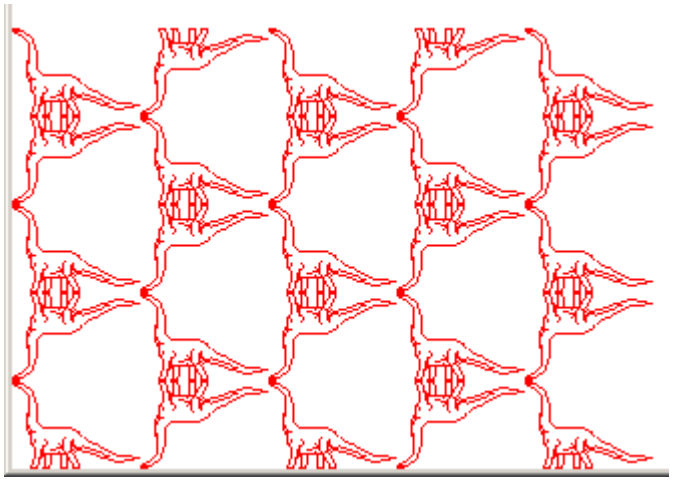

Figura 2 Copias de dinosaurio al posicionar diferentes puertos de vista y y girar la ventana

NOTA: En este caso en la ventana se va alternando el extremo superior e inferior. En cada iteración se intercambian los argumentos *top* y *bottom* de la función `setWindow()`.
Parecería más fácil invertir el PV, pero OpenGL no permite un PV que tenga un valor negativo de *height*.

## 3.4 Mostrar diferentes vistas de la misma imagen en diferentes puertos de vista

En una ventana de 640x480, representar diferentes vistas de la misma escena, de tal forma que los puertos de vista dividan en cuatro rectángulos iguales a dicha ventana. Estos cuatro puertos de vista estarán separados por líneas de otro color como se muestra en la figura 3.

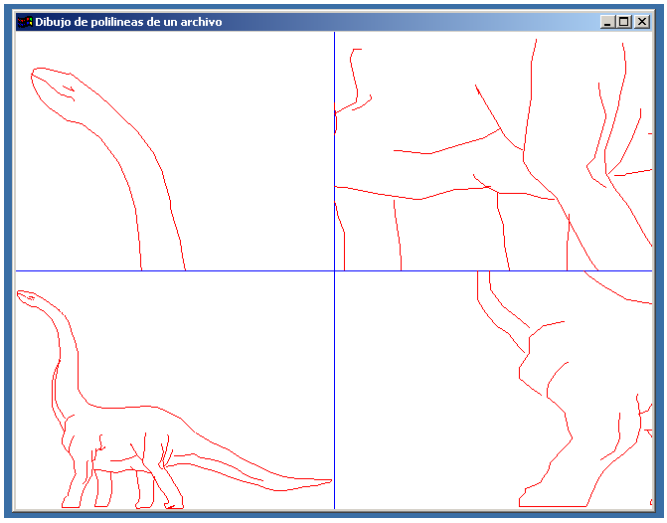


Figura 3 Distintas vista de la misma imegen en diferentes puertos de vista

## 3.5 Realizar una animación con diferentes zoom

Al ir variando la ventana (`setWindow`) y mostrarla en el mismo puerto de vista podemos realizar una animación de diferentes zoom.

A continuación se muestra el código

```
void myDisplay(void)
{
int NumFrames = 5;
float cx = 168.0, cy = 128.0; //Centra la ventana
float H = 320, W = 240.0f, aspect = 0.7f; //propiedades de la ventana
setViewport(0,640,0,480);
for (int frame = 0; frame < NumFrames; frame++) //para cada frame
    {
    glClear(GL_COLOR_BUFFER_BIT); //Limpia la pantalla
     W *= 0.7f; // reduce el ancho de ventana
     H = W * aspect; //mantiene la relación de aspecto
     setWindow(cx - W, cx + W,cy - H,cy + H); //selecc. siguiente ventana
     drawPolyLineFile("dino.dat"); //Dibuja el objeto
     glutSwapBuffers();
    }
 }
```

En este caso hay que utilizar un *"double buffering"*, para evitar el tiempo que tarda en mostrar la siguiente figura, de esta forma se muestra inmediatamente la siguiente figura.
En OpenGL el comando `glutSwapBuffers()`, produce que la imagen del buffer se transfiera a la ventana de la pantalla y sea visible por el usuario. Para hacer la reserva del buffer hay que emplear `GLUT_DOUBLE` en vez de `GLUT_SINGLE` en la rutina `main`.

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); //Selecciona el modo de pantalla
```

El comando `glutSwapBuffers();`debe ir directamente después de la función de dibujo de objeto de esta forma aunque tome tiempo en representar la imagen, produce un efecto más suave en la visualización.

## 3.6 Redimensionamiento de la ventana

Represente la figura del archivo "*dino.dat*" en una ventana inicial de 640x480 de tal forma que permita el redimensionamiento de ésta con el ratón. (Ver Cap. 3 Apartado 6, pág. 41-42 Redimensionando la ventana; redimensionando el evento)

## 3.7 Realizar un programa para representar las siguientes curvas dadas en paramétricas:

| | |
|---|---|
| Espiral | $r = a.\theta$ |
| Limason | $r = a.\cos\theta + b$ |
| Cardioide | $r = a.(1+\cos\theta)$ |
| Trifolio | $r = a.\cos(3.\theta)$ |
| Tetrafolio | $r = a.\cos(2.\theta)$ |

representar las curvas en una sola ventana, teniendo colores distintos cada curva.
Recordar que:

$$x = r\cos\theta$$
$$y = rsen\theta$$

Los valores del parámetro ***a*** para este caso vendrán definido por
a[5]= { {5.0f},{20.0f},{30.0f},{30.0f},{40.0f}};

Para centrar la figura desplazar las curvas los siguientes valores
centro[5]={ {-70.0f,-30.0f},{-8.0f,0.0f},{40.0f,-30.0f},{55.0f,50.0f},{-50.0f,45.0f}};

$$x = centro[i].x + r\cos\theta$$
$$y = centro[i].y + rsen\theta$$

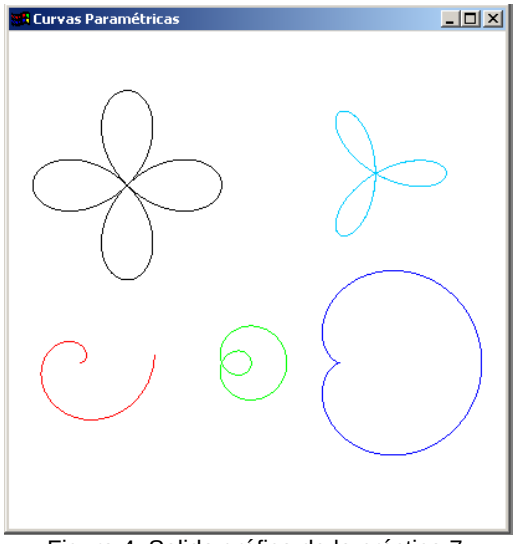


Figura 4. Salida gráfica de la práctica 7.

## 3.8 Realizar un programa para representar la animación de un punto

a) El programa representará un punto animado que siga el camino definido por la parábola de ecuación $y = 15\sqrt{x+50};\quad 0 \le x \le 400$. Representar asímismo los ejes coordenados y la parábola para el intervalo indicado. El movimiento será continuo, para el valor de x entre 0 y 400.

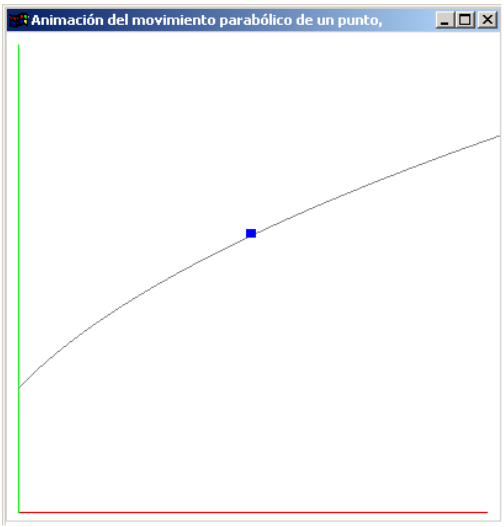


Figura 4. Salida gráfica de la práctica 8.

3.9 Realizar un programa para representar la animación de un punto continuo
Idem para el caso del tetrafolio definido en la práctica 3.7

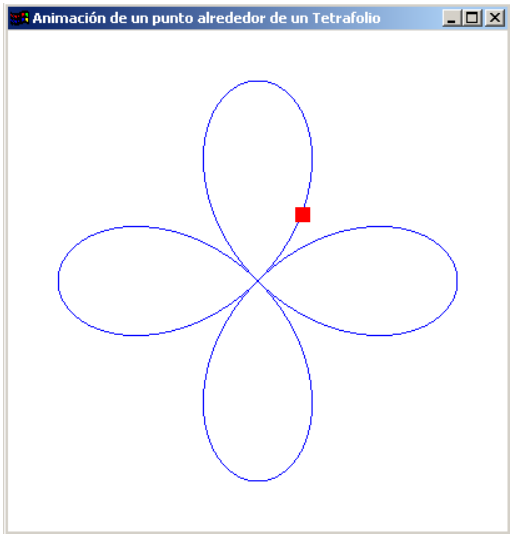


Figura 5. Salida gráfica de la práctica 9.