

Primero debemos comprender

- 1) Que comando saca el seno
- 2) Como saca el seno
- 3) Como graficar los valores
- 4) Como adecuarlo a -1 a 1

1) Comando que calcula la senoidal: $\sin(x)$

X debe ser expresada en radianes, Y se debe guardar en un registro de tipo real doble(o double)

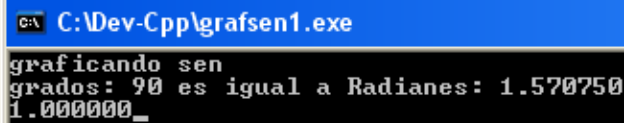
1.1) ¿Cómo convertir grados a radianes ?

Se sabe que $PI=180$ grados, o sea $PI(\text{radianes}) = 180(\text{grados})$ regla de tres si tengo de dato los grados y deseo obtener los radianes $\Rightarrow \text{radianes} = \text{grados} * PI / 180$

2. Y saquemos el seno

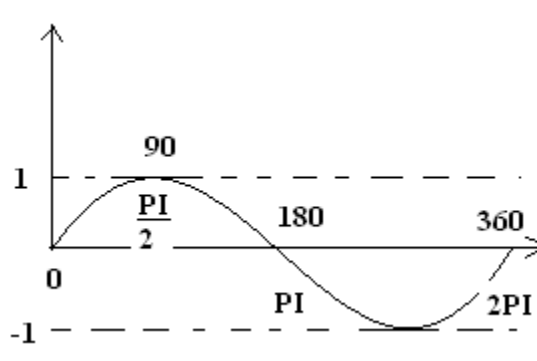
Realicemos primero el programa que convierte de grados a radianes

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define PI 3.1415
int grados;
float radianes;
double s;
main()
{
    printf("graficando sen");
    grados=90;
    //ec. para convertir 180 grados=PI radianes
    radianes=grados*PI/180;
    printf("\ngrados: %d es igual a Radianes: %f",grados,radianes);
    s=sin(radianes);
    printf("\n%lf",s);
    getch();
}
```



```
C:\Dev-Cpp\grafen1.exe
graficando sen
grados: 90 es igual a Radianes: 1.570750
1.000000_
```

2.1 Se comprende estos datos con la grafica siguiente:



Si pido el seno(90)=> sen(PI/2) y su resultado da =1 como la grafica. Cópialo, córrelo y verifica si para el resto de los valores funciona este programa.

Ya sabemos obtener los datos del seno.

2.2) utilicemos un proceso en repetición para obtener varios valores. UN FOR

QUE COMIENZE EN GRADOS=0, QUE EJECUTE EL PROCESO MIENTRAS GRADOS SEA MENOR QUE 361 Y QUE INCREMENTE UNO A UNO. Veamos el código. Corranlo.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define PI 3.1415
int grados;
float radianes;
double s;
main()
{
    printf("graficando sen");
    //grados=90;
    for (grados=0;grados<361;grados++)
    {
        radianes=grados*PI/180;
        s=sin(radianes);
        printf("\n%lf",s);
    }
    getch();
}
```

```
0.788108
0.777246
0.766147
0.754815
0.743252
0.731464
0.719452
0.707222
0.694776
0.682118
0.669252
0.656183
0.642914
0.629449
0.615792
0.601948
0.587920
0.573713
0.559332
0.544780
0.530062
0.515183
0.500147
0.484959
0.469623
0.454143
0.438526
0.422774
0.406895
0.390891
```

3) Graficar lo valores.

Ya tenemos listos los valores, ya se comprendió como utilizar el comando sin()
 Ahora debemos emparar datos entre -1 a 1 en horizontal y vertical como se requiere en devc gráficos.

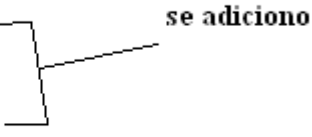
El resultado "s" ya estan entre -1 y 1

3.1) abrimos un nuevo proyecto en multimedia con opengl.

Quitamos lo de siempre y copiamos nuestro programa de aquí en partes

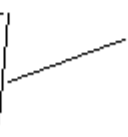
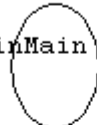
En la parte de cabeceras pasamos las cabeceras que utilizamos.

```
*****/  
  
#include <windows.h>  
#include <gl/gl.h>  
#include <conio.h>  
#include <math.h>  
#define PI 3.1415  
#include <stdio.h>  
  
/*****  
 * Function Declarations  
 *  
*****/
```



Las variables que declaramos las pegamos despues de MAIN y la llave {

```
/*****  
 * WinMain  
 *  
*****/  
  
int WINAPI WinMain (HINSTANCE hInstance,  
                   HINSTANCE hPrevInstance,  
                   LPSTR lpCmdLine,  
                   int iCmdShow)  
{  
    WNDCLASS wc;  
    HWND hWnd;  
    HDC hDC;  
    HGLRC hRC;  
    MSG msg;  
    BOOL bQuit = FALSE;  
    float theta = 0.0f;  
    int grados;  
    float radianes;  
    double s;  
}
```



Y finalmente nuestro código que calcula el seno en

```

        /* OpenGL animation code goes here */

        glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
        glClear (GL_COLOR_BUFFER_BIT);
        glPushMatrix ();
for (grados=0;grados<361;grados++)
    {
        radianes=grados*PI/180;
        s=sin(radianes);
        glBegin (GL_POINTS);
        glColor3f (1.0f, 0.0f, 0.0f);
        glVertex2f (radianes/(2*PI), s);
        glEnd ();
    }

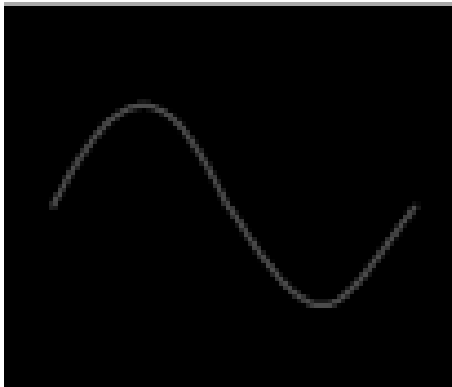
        glPopMatrix ();

        SwapBuffers (hDC);

        Sleep (2000);

```

adentro se inserto nuestro codigo



FIN

CODIGO COMPLETO*** Seleccionalo y copialo; sustituyelo por el codigo en tu proyecto.**

```

#include <windows.h>

#include <gl/gl.h>

#include <conio.h>

#include <math.h>

#define PI 3.1415

#include <stdio.h>

```

```
/******
```

```
* Function Declarations
```

```
*
```

```
*****/
```

```
LRESULT CALLBACK WndProc (HWND hWnd, UINT message,
```

```
WPARAM wParam, LPARAM lParam);
```

```
void EnableOpenGL (HWND hWnd, HDC *hDC, HGLRC *hRC);
```

```
void DisableOpenGL (HWND hWnd, HDC hDC, HGLRC hRC);
```

```
/******
```

```
* WinMain
```

```
*
```

```
*****/
```

```
int WINAPI WinMain (HINSTANCE hInstance,
```

```
                  HINSTANCE hPrevInstance,
```

```
                  LPSTR lpCmdLine,
```

```
                  int iCmdShow)
```

```
{
```

```
    WNDCLASS wc;
```

```
    HWND hWnd;
```

```
    HDC hDC;
```

```
    HGLRC hRC;
```

```
    MSG msg;
```

```
BOOL bQuit = FALSE;

float theta = 0.0f;

int grados;

float radianes;

double s;

/* register window class */

wc.style = CS_OWNDNC;

wc.lpfWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon (NULL, IDI_APPLICATION);

wc.hCursor = LoadCursor (NULL, IDC_ARROW);

wc.hbrBackground = (HBRUSH) GetStockObject (BLACK_BRUSH);

wc.lpszMenuName = NULL;

wc.lpszClassName = "GLSample";

RegisterClass (&wc);

/* create main window */

hWnd = CreateWindow (

    "GLSample", "OpenGL Sample",

    WS_CAPTION | WS_POPUPWINDOW | WS_VISIBLE,

    0, 0, 256, 256,

    NULL, NULL, hInstance, NULL);

/* enable OpenGL for the window */
```

```
EnableOpenGL (hWnd, &hDC, &hRC);
```

```
/* OpenGL animation code goes here */
```

```
glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
```

```
glClear (GL_COLOR_BUFFER_BIT);
```

```
glPushMatrix ();
```

```
for (grados=0;grados<361;grados++)
```

```
{
```

```
    radianes=grados*PI/180;
```

```
    s=sin(radianes);
```

```
        glBegin (GL_POINTS);
```

```
        glColor3f (1.0f, 0.0f, 0.0f);
```

```
        glVertex2f (radianes/(2*PI), s);
```

```
        glEnd ();
```

```
}
```

```
glPopMatrix ();
```

```
SwapBuffers (hDC);
```

```
Sleep (2000);
```

```

/* shutdown OpenGL */
DisableOpenGL (hWnd, hDC, hRC);

/* destroy the window explicitly */
DestroyWindow (hWnd);

return msg.wParam;
}

/*****
* Window Procedure
*
*****/

LRESULT CALLBACK WndProc (HWND hWnd, UINT message,
                          WPARAM wParam, LPARAM lParam)
{

switch (message)
{
case WM_CREATE:
    return 0;

case WM_CLOSE:
    PostQuitMessage (0);
    return 0;

```

```
case WM_DESTROY:
    return 0;

case WM_KEYDOWN:
    switch (wParam)
    {
    case VK_ESCAPE:
        PostQuitMessage(0);
        return 0;
    }
    return 0;

default:
    return DefWindowProc (hWnd, message, wParam, lParam);
}
}
```

```
/*  
*****  
*/
```

```
* Enable OpenGL
```

```
*
```

```
*****  
*/
```

```
void EnableOpenGL (HWND hWnd, HDC *hDC, HGLRC *hRC)
```

```
{
```

```
    PIXELFORMATDESCRIPTOR pfd;
```

```
    int iFormat;
```

```

/* get the device context (DC) */
*hDC = GetDC (hWnd);

/* set the pixel format for the DC */
ZeroMemory (&pfd, sizeof (pfd));
pfd.nSize = sizeof (pfd);
pfd.nVersion = 1;
pfd.dwFlags = PFD_DRAW_TO_WINDOW |
    PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
pfd.iPixelFormat = PFD_TYPE_RGBA;
pfd.cColorBits = 24;
pfd.cDepthBits = 16;
pfd.iLayerType = PFD_MAIN_PLANE;
iFormat = ChoosePixelFormat (*hDC, &pfd);
SetPixelFormat (*hDC, iFormat, &pfd);

/* create and enable the render context (RC) */
*hRC = wglCreateContext( *hDC );
wglMakeCurrent( *hDC, *hRC );

}

/*****

* Disable OpenGL

*

```

```
*****/
```

```
void DisableOpenGL (HWND hWnd, HDC hDC, HGLRC hRC)
```

```
{
```

```
    wglMakeCurrent (NULL, NULL);
```

```
    wglDeleteContext (hRC);
```

```
    ReleaseDC (hWnd, hDC);
```

```
}
```