

Programa que grafica el seno.

```
        glClear (GL_COLOR_BUFFER_BIT);
        glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
        glPushMatrix();
for (g=0;g<361;g++)
{

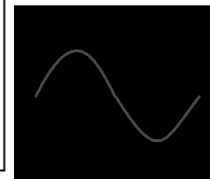
        x=g*PI/180;
        y=sin(x);

        glPointSize(3);
        glBegin (GL_POINTS);
        glColor3f (1.0f, 0.0f, 0.0f);
        glVertex2f (x/(2*PI), y);
        glEnd ();

}

        glPopMatrix ();

        SwapBuffers (hDC);
        Sleep (2000);
```



Pero si se requiere pasar un punto por esta trayectoria entonces se cambia así

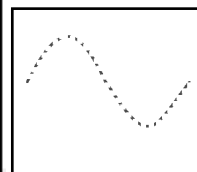
```
        glClearColor (1.0f, 1.0f, 1.0f, 1.0f);
for (g=0;g<361;g++)
{

        glPushMatrix();
        x=g*PI/180;
        y=sin(x);
        glClear (GL_COLOR_BUFFER_BIT);
        glPointSize(3);
        glBegin (GL_POINTS);
        glColor3f (1.0f, 0.0f, 0.0f);
        glVertex2f (x/(2*PI), y);
        glEnd ();

        glPopMatrix ();

        SwapBuffers (hDC);
        Sleep (20);

}
```



```
//programa completo1
```

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

```
* Includes
```

```
*
```

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

```
#include <windows.h>
```

```
#include <gl/gl.h>
```

```
#include <math.h>
```

```
#define PI 3.1415
```

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

```
* 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 g;

    double x,y;

    /* register window class */

    wc.style = CS_OWNDC;

    wc.lpfnWndProc = 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 */  
  
glClear (GL_COLOR_BUFFER_BIT);  
glClearColor (0.0f, 0.0f, 0.0f, 0.0f);  
glPushMatrix();  
for(g=0;g<361;g++)  
{  
  
    x=g*PI/180;  
    y=sin(x);  
  
    glPointSize(3);  
    glBegin (GL_POINTS);
```

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

        glVertex2f (x/(2*PI), y);

        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, &pfid);

/* 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);
}

```