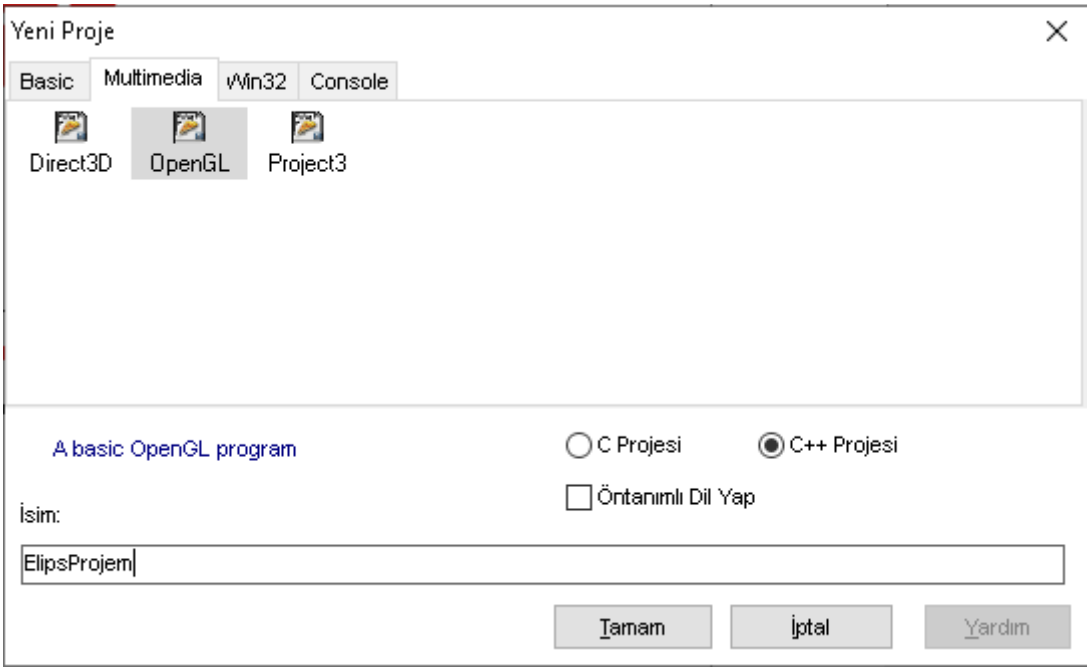
DEVC++ TA OPENGL İLE ELİPS ÇİZME ÖRNEĞİ

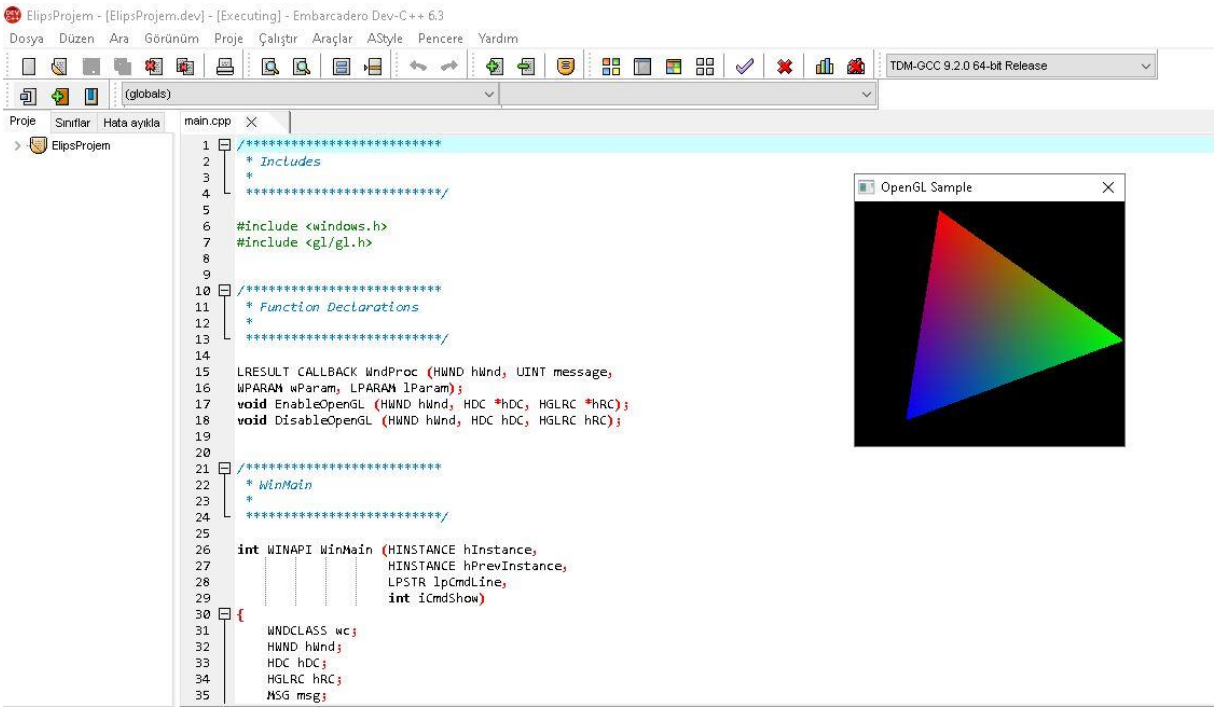
ÖĞR. GÖR. UFUK ŞANVER

Öncelikle Dev C++ ta Örnek→ Yeni Proje → Multimedya→ OpenGL i seçiyoruz



Tamam ve İleri diyerek projemizi oluşturuyoruz.

Ardından gelen hazır örnek dosyayı koşalım. Derle ve Çalıştır dediğimizde karşımıza dönen bir üçgen programı çıkacaktır.



Dev C++ da OpenGL in varsayımsal olarak kullandığı koordinat sistemi [-1,+1] koordinat sistemidir. Bu sistemde ekranın merkezi 0,0 koordinatı olur. Sol kenar x=-1, sağ kenar x=+1, üst kenar y=+1, alt kenar y=-1olur.

Bu nedenle piksel koordinatlarına göre yazılacak programlarda koordinatların [-1,+1] aralığına indirgenmesi gerekmektedir.

Ekran penceresi büyüklüğü için en ve boy sabit olarak tanımlansın.

const float en=600.0;

const float boy=600.0;

En ve boy değerlerine uygun olarak piksel koordinatlarını -1- +1 aralığına iz düşürmek için bir fonksiyon tanımlansın. Burada 1 - (-1) = 2 olduğuna göre verilen en ve boyun 2'ye iz düşürülmesi gerekir. Bu amaçla en ve boy koordinatlarına yönelik birer fonksiyon oluşturalım.

```
float endonustur(float xkoor){
        return (xkoor*2)/en - 1.0;
}
float boydonustur(float ykoor){
        return (ykoor*2)/boy - 1.0;
}
```

Ardından üçgen çizen fonksiyonu kaldırıp kırmızı renkte elips çizen fonksiyonumuzu yazalım.

Elips çiziminde elipsin parametrik ifadesinden yararlanılsın.

X= Xmerkez + r1cos(Θ)

Y= Ymerkez + r2sin(Θ)

Deneme kodunda üçgen çizen kod parçasını çıkartıp elips kodunu yerleştirelim.

Burada açıları radyan cinsinden verilmelidir. Bu amaçla PI tanımlansın ve dereceden radyana çeviren bir fonksiyon tanımlansın.

Çizilmek istenen elipsin Merkezi 200,200 ve yarıçapları 100,50 olsun.

```
  float r1=100.0;
  float r2=50.0;
  float xmerkez=200.0;
  float ymerkez=200.0;
 float derece=0.0;
```

Kırmızı elips çizen ifade:

```
        glColor3f (1.0f, 0.0f, 0.0f);
        glBegin (GL_LINE_LOOP);
 for(derece=0.0;derece<360.0;derece+=5.0){
glVertex2f(endonustur(xmerkez+r1*cos(radyan(derece))),boydonustur(ymerkez+r2*sin(radyan(derece))));
```

```
}
        glEnd ();
```

DEVC++ ORJİNAL OPENGL AÇILIŞ PROGRAMI MODİFİYE EDİLEREK YAPILAN ELİPS PROGRAMI

/*DEV C++ IN BAŞLANGIÇ PROGRAMI MODİFİYE EDİLEREK ELDE EDİLMİŞTİR*/

```
/*************************
 * Includes
 *
 *************************/

#include <windows.h>
#include <gl/gl.h>
#include <math.h>
#define PI 3.145

const float en=600.0;
const float boy=600.0;

float endonustur(float xkoor){
	return (xkoor*2)/en - 1.0;
}

float boydonustur(float ykoor){
	return (ykoor*2)/boy - 1.0;
}

float radyan(float derece){
	return (derece*PI)/180.0;
}
/*************************
 * 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)
{
  float r1=100.0;
  float r2=50.0;
  float xmerkez=200.0;
  float ymerkez=200.0;
  float derece=0.0;


   WNDCLASS wc;
   HWND hWnd;
   HDC hDC;
   HGLRC hRC;
   MSG msg;
```

```
BOOL bQuit = FALSE;
float theta = 0.0f;



/* 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", "ELIPS CIZ",
  WS_CAPTION | WS_POPUPWINDOW | WS_VISIBLE,
  0, 0, en, boy,
  NULL, NULL, hInstance, NULL);

/* enable OpenGL for the window */
EnableOpenGL (hWnd, &hDC, &hRC);

/* program main loop */
while (!bQuit)
{
```

```
    /* check for messages */
    if (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
    {
      /* handle or dispatch messages */
      if (msg.message == WM_QUIT)
      {
        bQuit = TRUE;
      }
      else
      {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
      }
    }
    else
    {
      /* OpenGL animation code goes here */

      glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
      glClear (GL_COLOR_BUFFER_BIT);

      glPushMatrix ();
     // glRotatef (theta, 0.0f, 0.0f, 1.0f);
      glColor3f (1.0f, 0.0f, 0.0f);
      glBegin (GL_LINE_LOOP);
      for(derece=0.0;derece<360.0;derece+=5.0){
                        glVertex2f
(endonustur(xmerkez+r1*cos(radyan(derece))),boydonustur(ymerkez+r2*sin(radyan(derece))));
                        }

      glEnd ();
```

```
            glPopMatrix ();

            SwapBuffers (hDC);

           // theta += 1.0f;
            Sleep (1);
         }
      }

      /* 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.iPixelType = 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);
}
```