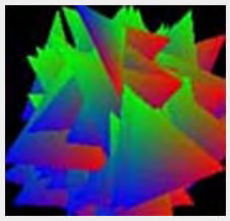
Course 3D_OpenGL: 3D-Graphics with C++ and OpenGL
C1: The Complete Code of Moving Triangles

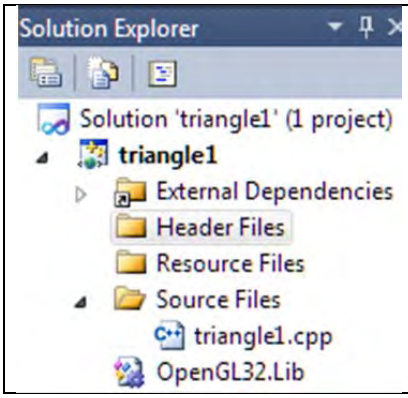


This project is the OpenGL-clone of
3D-Computer Graphics with Managed DirectX, Chapter C1: Moving Triangles and of
3D-Computer Graphics with XNA, Chapter C1: Moving Triangles.

Start Visual C++ 2010 Express:
`File` → `New` → `Project` → `Win32 Project` → `Name: triangle1` → `Location: C:\temp` →
uncheck `Create directory for solution` → `Win 32 Application Wizard - triangle1` →
`Application Settings` → `Application type: Windows application` →
`Additional options: Empty project` → quit with button `Finish`.

In the `Solution Explorer -` window **Right**-click `triangle1`.
A drop-down menu appears with a branch: "`Add`". Click `Add` → `Existing item...` →
`C:\Program Files\Microsoft SDKs\Windows\v7.0A\Lib\OpenGl32.Lib` → Quit with button `Add`.
(If you don't find `OpenGl32.Lib` download **opengl32.lib** and **gl.h**.)
Check whether `OpenGl32.Lib` arrived in the Solution Explorer tree.

In the `Solution Explorer -` window click the + sign in front of `triangle1`.
A tree appears with a branch: "`Source Files`". **Right**-click `Source Files`. A drop-down menu appears.
Click `Add` → `New Item...` → `Add New item - triangle1` →
`Visual Studio installed templates: C++ File (.cpp)` → `Name: triangle1` →
`Location: c:\Temp\triangle1` → Quit with button `Add`.
Check whether `triangle1.cpp` arrived under branch `Source Files` in the Solution Explorer tree.



If the `Solution Explorer -` window is invisible,
open it via the main menu: `View → Solution Explorer`.

Copy the following code into the empty source file `triangle1.cpp`:

```
#include <windows.h>
#include <time.h>
#include <C:\Program Files\Microsoft SDKs\Windows\v7.0A\include\gl\Gl.h> //adjust your path to Gl.h !

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
HWND     hWnd;
HDC      hDC;
HGLRC    hRC;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int iCmdShow)
{ const int nTriangles = 100;
  float dx[nTriangles];
  float dy[nTriangles];
  float dz[nTriangles];
  float ax[nTriangles];
  float ay[nTriangles];
  float az[nTriangles];
  srand( (unsigned)time(NULL) ); //initialize random number generator
  for ( int i=0; i < nTriangles; i++ )
  { dx[i] = -0.5f + (float)rand() / RAND_MAX; //something between -0.5 and +0.5
    dy[i] = -0.5f + (float)rand() / RAND_MAX; //something between -0.5 and +0.5
    dz[i] = -0.5f + (float)rand() / RAND_MAX; //something between -0.5 and +0.5
    ax[i] = 90.0f * (float)rand() / RAND_MAX; //something between  0.0 and 90.0
    ay[i] = 90.0f * (float)rand() / RAND_MAX; //something between  0.0 and 90.0
    az[i] = 90.0f * (float)rand() / RAND_MAX; //something between  0.0 and 90.0
  }
  WNDCLASS wc;
  wc.style         = CS_OWNDC;
  wc.style         = CS_HREDRAW | CS_VREDRAW;
  wc.lpfnWndProc   = WndProc;
  wc.cbClsExtra    = 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 = L"ogl1";
  RegisterClass( &wc );
  hWnd = CreateWindow( wc.lpszClassName, L"C++ OpenGL Chapter 1",
                       WS_CAPTION | WS_POPUPWINDOW | WS_VISIBLE,
                       0, 0, 600, 600, NULL, NULL, hInstance, NULL );
  hDC = GetDC( hWnd );
  PIXELFORMATDESCRIPTOR pfd;
  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;
  int format = ChoosePixelFormat( hDC, &pfd );
  SetPixelFormat( hDC, format, &pfd );
  hRC = wglCreateContext( hDC );
  wglMakeCurrent( hDC, hRC );
  GLfloat vertex[] = {-0.5f,-0.5f, 0.0f,  //lower left  vertex
                       0.0f, 0.5f, 0.0f,  //mid   peek  vertex
                       0.5f,-0.5f, 0.0f };//lower right vertex
  GLfloat color[]  = { 0.0f, 0.0f, 1.0f,  //blue
                       0.0f, 1.0f, 0.0f,  //green
                       1.0f, 0.0f, 0.0f };//red
  glVertexPointer( 3, GL_FLOAT, 0, vertex );
  glColorPointer ( 3, GL_FLOAT, 0, color );
  glEnableClientState( GL_VERTEX_ARRAY );
  glEnableClientState( GL_COLOR_ARRAY  );
```

```
  while ( TRUE ) //infinite loop
  { glClearColor( 0.0f, 0.0f, 0.0f, 0.0f );
    glClear( GL_COLOR_BUFFER_BIT );
    for ( int i=0; i < nTriangles; i++ )
    { glLoadIdentity(); //erase all former transforms
      //glScalef    ( 0.7f, 0.7f, 0.7f );
      //glTranslatef( dx[i], dy[i], dz[i] );
      glRotatef( ax[i]+=0.01f, 1.0f, 0.0f, 0.0f ); //around x-axis
      glRotatef( ay[i]+=0.01f, 0.0f, 1.0f, 0.0f ); //around y-axis
      glRotatef( az[i]+=0.01f, 0.0f, 0.0f, 1.0f ); //around z-axis
      glTranslatef( dx[i], dy[i], dz[i] );         //shifts
      glDrawArrays( GL_TRIANGLES, 0, 3 );          //draw it
    }
    SwapBuffers( hDC ); //show all
    MSG msg;            //listen to what happens outside the loop
    if ( PeekMessage( &msg, NULL, 0, 0, PM_REMOVE ) )
    { if ( msg.message == WM_QUIT ) return 0; //stop infinite loop
      TranslateMessage( &msg );
      DispatchMessage ( &msg );
    }
  } //end of infinite loop
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{ switch (message)
  { case WM_CLOSE: PostQuitMessage( 0 ); return 0; //send WM_QUIT to stop infinite loop
    case WM_QUIT : wglMakeCurrent( hDC, NULL );    //disconnect hRC from hDC
                   wglDeleteContext( hRC );        //say goodbye to hRC
                   ReleaseDC( hWnd, hDC );         //say goodbye to hDC
                   DestroyWindow( hWnd ); return 0;//say goodbye to hWnd
    default      : return DefWindowProc( hWnd, message, wParam, lParam );
  }
}
```