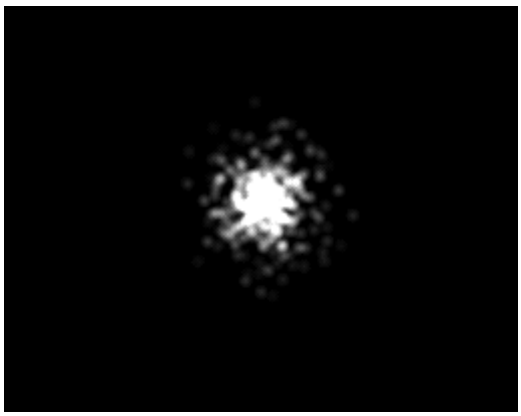
فصل بیست و هشتم

طرز استفاده از OpenGL در BC++ Builder و VC++ و اسمبلی



مقدمه :

با توجه به اینکه OpenGL به زبان C نوشته شده است و همچنین برای قابل استفاده تر شدن این کتاب برای سایر برنامه نویسان ، طرز استفاده از OpenGL در کامپایلرهای مختلف نیز بررسی می گردد .

طرز استفاده از OpenGL در VC++ :

پس از ایجاد یک Win32 Application جدید و نه console application در Visual C++ ، باید کتابخانه های OpenGL را به برنامه خود لینک کنید . برای انجام این کار در منوی Project ، قسمت Settings را انتخاب کرده و در فرم ظاهر شده ، بر روی قسمت LINK کلیک کنید(alt+F7->Link) . تحت قسمت Object/Library Modules ، در ابتدای خط ، قبل از kernel32.lib ، عبارت زیر را اضافه کنید :

OpenGL32.lib GLu32.lib GLaux.lib

پس از کلیک کردن بر روی OK ، آماده به نوشتن برنامه های OpenGL می باشید .

نکته :

اگر در هنگام کامپایل برنامه به error lnk2001 برخوردید ، عبارت subsystem:console را از قسمت Link یاد شده ، حذف کنید .

برنامه VC++ فصل :

```
/*
                Particle Engine tutorial
*/

// WINDOWS & OPENGL STUFF //
#include <math.h>                // Header File For Math functions
#include <windows.h>    // Header File For Windows
#include <stdio.h>               // Header File For Standard Input / Output
#include <gl\gl.h>               // Header File For The OpenGL32 Library
#include <gl\glaux.h>    // Header File For The Glaux Library

HDC             hDC=NULL;                       // Private GDI Device Context
HGLRC   hRC=NULL;                       // Permanent Rendering Context
HWND    hWnd=NULL;                      // Holds Our Window Handle
bool    keys[256];                      // Array Used For The Keyboard Routine
bool    active=TRUE;            // Window Active Flag Set To TRUE By Default
bool    fullscreen=TRUE;        // Fullscreen Flag Set To Fullscreen Mode By Default
LRESULT         CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  // Declaration For WndProc
///////////////////////////

#define MAX_PARTICLES 500 // # of particles

typedef struct                                          // Create A Structure For Particle
{
        float   life;                                   // Particle Life
        float   fade;                                   // Fade Speed
        float   x;                                              // X Position
        float   y;                                              // Y Position
        float   z;                                              // Z Position
        float   xi;                                             // X Direction
        float   yi;                                             // Y Direction
        float   zi;                                             // Z Direction
}
particles;                                                      // Particles Structure

particles particle[MAX_PARTICLES];      // Particle Array (Room For Particle Info)

int loop;
float V, Angle;

// This function will make the blurry texture perfect for the particles.
```

```
// Sorry for the lack of comments in it(some of it confuses me too).
int setCheckTexture(void)
{
   int texWidth = 256;
   int texHeight = 256;
   GLubyte *texPixels, *p;
   int texSize;
   int i, j;
   int radius;

   texSize = texWidth*texHeight*4*sizeof(GLubyte);
   texPixels = (GLubyte *) malloc(texSize);
   if (texPixels == NULL)
   {
         return false;
   }

   p = texPixels;
   for (i=0; i<texHeight; ++i)
   {
         for (j=0; j<texWidth; ++j)
    {
          GLuint dist = hypot(float(i - (texHeight / 2)),float(j - (texWidth / 2)));

      float color = 255-(dist*1.8);
      if (color < 0) color = 0;
      p[0] = color;
      p[1] = color;
      p[2] = color;
      p[3] = color;
      p+=4;
         }
   }

        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, texWidth, texHeight,
                0, GL_RGBA, GL_UNSIGNED_BYTE, texPixels);

   free(texPixels);

   return true;
}


// Resize And Initialize The GL Window
GLvoid ReSizeGLScene(GLsizei width, GLsizei height)
{
        if (height==0)                          // Prevent A Divide By Zero By
        {
                height=1;                       // Making Height Equal One
        }

        glViewport(0,0,width,height);     // Reset The Current Viewport

        glMatrixMode(GL_PROJECTION);            // Select The Projection Matrix
        glLoadIdentity();                       // Reset The Projection Matrix
```

```
        // Calculate The Aspect Ratio Of The Window
        gluPerspective(45.0f,(GLfloat)width/(GLfloat)height,0.1f,100.0f);

        glMatrixMode(GL_MODELVIEW);          // Select The Modelview Matrix
        glLoadIdentity();                    // Reset The Modelview Matrix
}


int InitGL(GLvoid)                           // All Setup For OpenGL Goes Here
{
   if (!setCheckTexture())
   {
     return false;
   }
        glLoadIdentity();                    // Reset The Projection Matrix
   glEnable(GL_TEXTURE_2D);
   glEnable(GL_BLEND);
   glShadeModel(GL_SMOOTH);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE);

   for (loop=0;loop<MAX_PARTICLES;loop++)   // Initializes All The Textures
        {
     particle[loop].life=1.0f; // Give All The Particles Full Life
     particle[loop].fade=float(rand()%100)/1000.0f+0.05f; // Random Fade Speed

     V = float(rand()%25);  // Speed of the particle
     Angle = float(rand()%360);  // Angle of the particle

     particle[loop].x = 0;  // Set X position
     particle[loop].y = 0;  // Set Y position
     particle[loop].z = 0;  // Set Z position

     particle[loop].xi = sin(Angle) * V;  // Set X velocity
     particle[loop].yi = cos(Angle) * V;  // Set Y velocity
     particle[loop].zi = float(((rand()%10)-5)/10) * V;  // Set Z velocity
        }

   return true;
}


int DrawGLScene(GLvoid)                      // Here's Where We Do All The Drawing
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glLoadIdentity();
  glTranslatef(0,0,-15);

  for (loop=0; loop<MAX_PARTICLES; loop++) // Loop Through All The Particles
  {
    float x=particle[loop].x; // Grab Our Particle X Position
    float y=particle[loop].y; // Grab Our Particle Y Position
    float z=particle[loop].z; // Grab Our Particle Z Position

    // Draw The Particle Using Our RGB Values, Fade The Particle Based On It's Life
    glColor4f(.5f,.5f,1.f,particle[loop].life);

    glBegin(GL_TRIANGLE_STRIP); // Build Quad From A Triangle Strip
```

```
    glTexCoord2f(1,1); glVertex3f(x+0.2f,y+0.2f,z); // Top Right
    glTexCoord2f(0,1); glVertex3f(x-0.2f,y+0.2f,z); // Bottom Right
    glTexCoord2f(1,0); glVertex3f(x+0.2f,y-0.2f,z); // Top Left
    glTexCoord2f(0,0); glVertex3f(x-0.2f,y-0.2f,z); // Bottom Left
    glEnd(); // Done Building Triangle Strip

    particle[loop].x+=particle[loop].xi/250;  // Move On The X Axis By X Speed
    particle[loop].y+=particle[loop].yi/250;  // Move On The Y Axis By Y Speed
    particle[loop].z+=particle[loop].zi/250;  // Move On The Z Axis By Z Speed

    // Slow down the particles
    particle[loop].xi*=.99;
    particle[loop].yi*=.99;
    particle[loop].zi*=.99;

    particle[loop].life-=particle[loop].fade; // Reduce Particles Life By 'Fade'

    if (particle[loop].life<0.05f) // If Particle Is Burned Out
    {
      particle[loop].life=1.0f; // Give It New Life
      particle[loop].fade=float(rand()%100)/10000 + 0.005f; // Random Fade Value
      particle[loop].x= 0; // Center On X Axis
      particle[loop].y= 0; // Center On Y Axis
      particle[loop].z= 0; // Center On Z Axis
      V = (float((rand()%9))+1);
      Angle = float(rand()%360);

      particle[loop].xi = sin(Angle) * V;
      particle[loop].yi = cos(Angle) * V;
      particle[loop].zi = ((rand()%10)-5)/5;
    }
  }

  return TRUE;                                  // Keep Going
}

GLvoid KillGLWindow(GLvoid)                     // Properly Kill The Window
{
        if (hRC)                                // Do We Have A Rendering Context?
        {
                // Are We Able To Release The DC And RC Contexts?
                if (!wglMakeCurrent(NULL,NULL))
                {
                        MessageBox(NULL,"Release Of DC And RC Failed.",
                                "SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
                }

                if (!wglDeleteContext(hRC))     // Are We Able To Delete The RC?
                {
                        MessageBox(NULL,"Release Rendering Context Failed.",
                                "SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
                }
                hRC=NULL;                       // Set RC To NULL
        }

        if (hDC && !ReleaseDC(hWnd,hDC))        // Are We Able To Release The DC
        {
                MessageBox(NULL,"Release Device Context Failed.",
```

```
                "SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
            hDC=NULL;                       // Set DC To NULL
        }

        if (hWnd && !DestroyWindow(hWnd))    // Are We Able To Destroy The Window?
        {
            MessageBox(NULL,"Could Not Release hWnd.",
                "SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
            hWnd=NULL;                      // Set hWnd To NULL
        }

        if (fullscreen)                     // Are We In Fullscreen Mode?
        {
            ChangeDisplaySettings(NULL,0); // If So Switch Back To The Desktop
            ShowCursor(TRUE);               // Show Mouse Pointer
        }
}

/*      This Code Creates Our OpenGL Window.  Parameters Are:
        title                   - Title To Appear At The Top Of The Window
 *      width                   - Width Of The GL Window Or Fullscreen Mode
 *      height                  - Height Of The GL Window Or Fullscreen Mode        *
 *      bits                    - Number Of Bits To Use For Color (8/16/24/32)      *
 *      fullscreenflag  - Use Fullscreen Mode (TRUE) Or Windowed Mode (FALSE)       */

BOOL CreateGLWindow(char* title, int width, int height, int bits, bool fullscreenflag)
{
        GLuint          PixelFormat;            // Holds The Results After Searching For A Match
        HINSTANCE       hInstance;              // Holds The Instance Of The Application
        WNDCLASS        wc;                     // Windows Class Structure
        DWORD           dwExStyle;              // Window Extended Style
        DWORD           dwStyle;                // Window Style

        fullscreen=fullscreenflag;              // Set The Global Fullscreen Flag

        // Grab An Instance For Our Window
        hInstance                       = GetModuleHandle(NULL);
        // Redraw On Size, And Own DC For Window.
        wc.style                = CS_HREDRAW | CS_VREDRAW | CS_OWNDC;
        wc.lpfnWndProc          = (WNDPROC) WndProc; // WndProc Handles Messages
        wc.cbClsExtra           = 0;                    // No Extra Window Data
        wc.cbWndExtra           = 0;                    // No Extra Window Data
        wc.hInstance            = hInstance;            // Set The Instance
        wc.hIcon                = LoadIcon(NULL, IDI_WINLOGO); // Load The Default Icon
        wc.hCursor              = LoadCursor(NULL, IDC_ARROW);// Load The Arrow Pointer
        wc.hbrBackground        = NULL;         // No Background Required For GL
        wc.lpszMenuName                 = NULL;
        // We Don't Want A Menu
        wc.lpszClassName        = "OpenGL";             // Set The Class Name

        if (!RegisterClass(&wc))        // Attempt To Register The Window Class
        {
            MessageBox(NULL,"Failed To Register The Window Class.",
                "ERROR",MB_OK|MB_ICONEXCLAMATION);
            return FALSE;                       // Return FALSE
        }

        if (fullscreen)                         // Attempt Fullscreen Mode?
```

```
        {
        DEVMODE dmScreenSettings;               // Device Mode
        memset(&dmScreenSettings,0,sizeof(dmScreenSettings)); // Makes Sure Memory's Cleared
        dmScreenSettings.dmSize=sizeof(dmScreenSettings);// Size Of The Devmode Structure
        dmScreenSettings.dmPelsWidth  = width;                  // Selected Screen Width
                dmScreenSettings.dmPelsHeight = height;         // Selected Screen Height
                dmScreenSettings.dmBitsPerPel  = bits;          // Selected Bits Per Pixel
                dmScreenSettings.dmFields=DM_BITSPERPEL|DM_PELSWIDTH|DM_PELSHEIGHT;

// Try To Set Selected Mode And Get Results.  NOTE: CDS_FULLSCREEN Gets Rid Of Start Bar.
                if
(ChangeDisplaySettings(&dmScreenSettings,CDS_FULLSCREEN)!=DISP_CHANGE_SUCCESSFUL)
                {
                        // If The Mode Fails, Offer Two Options.  Quit Or Use Windowed Mode.
                        if (MessageBox(NULL,
                                " Use Windowed Mode Instead?",
                                "GL",MB_YESNO|MB_ICONEXCLAMATION)==IDYES)
                        {
                        fullscreen=FALSE; // Windowed Mode Selected.  Fullscreen = FALSE
                        }
                        else
                        {
        // Pop Up A Message Box Letting User Know The Program Is Closing.
        MessageBox(NULL,"Program Will Now Close.","ERROR",MB_OK|MB_ICONSTOP);
                                return FALSE;
                // Return FALSE
                        }
                }
        }
        if (fullscreen)             // Are We Still In Fullscreen Mode?
        {
                dwExStyle=WS_EX_APPWINDOW;                  // Window Extended Style
        dwStyle=WS_POPUP | WS_CLIPSIBLINGS | WS_CLIPCHILDREN; // Windows Style
                ShowCursor(FALSE);              // Hide Mouse Pointer
        }
        else
        {
dwExStyle=WS_EX_APPWINDOW | WS_EX_WINDOWEDGE;       // Window Extended Style
dwStyle=WS_OVERLAPPEDWINDOW | WS_CLIPSIBLINGS | WS_CLIPCHILDREN; // Windows Style
        }

        // Create The Window
        if (!(hWnd=CreateWindowEx(    dwExStyle,       // Extended Style For The Window
                "OpenGL",                       // Class Name
                title,                          // Window Title
                dwStyle,                        // Window Style
                0, 0,                           // Window Position
                width, height,              // Selected Width And Height
                NULL,                           // No Parent Window
                NULL,                           // No Menu
                hInstance,                      // Instance
                NULL)))                         // Dont Pass Anything To WM_CREATE
        {
                KillGLWindow();                         // Reset The Display
                MessageBox(NULL,"Window Creation
Error.","ERROR",MB_OK|MB_ICONEXCLAMATION);
                return FALSE;                   // Return FALSE
        }
```

```
static	PIXELFORMATDESCRIPTOR pfd=// pfd Tells Windows How We Want Things To Be
	{
		sizeof(PIXELFORMATDESCRIPTOR),	// Size Of This Pixel Format Descriptor
		1,	// Version Number
		PFD_DRAW_TO_WINDOW |	// Format Must Support Window
		PFD_SUPPORT_OPENGL |	// Format Must Support OpenGL
		PFD_DOUBLEBUFFER,	// Must Support Double Buffering
		PFD_TYPE_RGBA,	// Request An RGBA Format
		bits,	// Select Our Color Depth
		0, 0, 0, 0, 0, 0,	// Color Bits Ignored
		0,	// No Alpha Buffer
		0,	// Shift Bit Ignored
		0,	// No Accumulation Buffer
		0, 0, 0, 0,	// Accumulation Bits Ignored
		16,	// 16Bit Z-Buffer (Depth Buffer)
		0,	// No Stencil Buffer
		0,	// No Auxiliary Buffer
		PFD_MAIN_PLANE,	// Main Drawing Layer
		0,	// Reserved
		0, 0, 0	// Layer Masks Ignored
	};

	if (!(hDC=GetDC(hWnd)))	// Did We Get A Device Context?
	{
		KillGLWindow();	// Reset The Display
		MessageBox(NULL,"Can't Create A GL Device Context.",
			"ERROR",MB_OK|MB_ICONEXCLAMATION);
		return FALSE;	// Return FALSE
	}

if (!(PixelFormat=ChoosePixelFormat(hDC,&pfd))// Did Windows Find A Matching Pixel Format?
	{
		KillGLWindow();	// Reset The Display
		MessageBox(NULL,"Can't Find A Suitable PixelFormat.",
			"ERROR",MB_OK|MB_ICONEXCLAMATION);
		return FALSE;	// Return FALSE
	}

	if(!SetPixelFormat(hDC,PixelFormat,&pfd))// Are We Able To Set The Pixel Format?
	{
		KillGLWindow();	// Reset The Display
		MessageBox(NULL,"Can't Set The PixelFormat.",
			"ERROR",MB_OK|MB_ICONEXCLAMATION);
		return FALSE;	// Return FALSE
	}

	if (!(hRC=wglCreateContext(hDC)))// Are We Able To Get A Rendering Context?
	{
		KillGLWindow();	// Reset The Display
		MessageBox(NULL,"Can't Create A GL Rendering Context.",
			"ERROR",MB_OK|MB_ICONEXCLAMATION);
		return FALSE;	// Return FALSE
	}

	if(!wglMakeCurrent(hDC,hRC))	// Try To Activate The Rendering Context
	{
		KillGLWindow();	// Reset The Display
```

```
            MessageBox(NULL,"Can't Activate The GL Rendering Context.","ERROR",
                MB_OK|MB_ICONEXCLAMATION);
            return FALSE;                   // Return FALSE
        }
        ShowWindow(hWnd,SW_SHOW);           // Show The Window
        SetForegroundWindow(hWnd);          // Slightly Higher Priority
        SetFocus(hWnd);                     // Sets Keyboard Focus To The Window
        ReSizeGLScene(width, height);       // Set Up Our Perspective GL Screen

        if (!InitGL())                  // Initialize Our Newly Created GL Window
        {
            KillGLWindow();             // Reset The Display
            MessageBox(NULL,"Initialization Failed.",
                "ERROR",MB_OK|MB_ICONEXCLAMATION);
            return FALSE;               // Return FALSE
        }

        return TRUE;                    // Success
}

LRESULT CALLBACK WndProc(   HWND  hWnd,  // Handle For This Window
        UINT    uMsg,               // Message For This Window
        WPARAM      wParam,         // Additional Message Information
        LPARAM      lParam)         // Additional Message Information
{
        switch (uMsg)                   // Check For Windows Messages
        {
            case WM_ACTIVATE:           // Watch For Window Activate Message
            {
                if (!HIWORD(wParam))  // Check Minimization State
                {
                    active=TRUE;    // Program Is Active
                }
                else
                {
                    active=FALSE;   // Program Is No Longer Active
                }

                return 0;       // Return To The Message Loop
            }
            case WM_SYSCOMMAND:     // Intercept System Commands
            {
                switch (wParam)     // Check System Calls
                {
                case SC_SCREENSAVE:  // Screensaver Trying To Start?
                case SC_MONITORPOWER:       // Monitor Trying To Enter Powersave?
                return 0;           // Prevent From Happening
                }
                break;              // Exit
            }

            case WM_CLOSE:              // Did We Receive A Close Message?
            {
                PostQuitMessage(0);     // Send A Quit Message
                return 0;               // Jump Back
            }
            case WM_KEYDOWN:            // Is A Key Being Held Down?
            {
```

```
                keys[wParam] = TRUE; // If So, Mark It As TRUE
                return 0;            // Jump Back
            }

            case WM_KEYUP:           // Has A Key Been Released?
            {
                keys[wParam] = FALSE;// If So, Mark It As FALSE
                return 0;            // Jump Back
            }
            case WM_SIZE:            // Resize The OpenGL Window
            {
        ReSizeGLScene(LOWORD(lParam),HIWORD(lParam));  // LoWord=Width, HiWord=Height
                return 0;                    // Jump Back
            }
        }

        // Pass All Unhandled Messages To DefWindowProc
        return DefWindowProc(hWnd,uMsg,wParam,lParam);
}


int WINAPI WinMain(  HINSTANCE   hInstance,      // Instance
            HINSTANCE   hPrevInstance,           // Previous Instance
            LPSTR       lpCmdLine,               // Command Line Parameters
            int             nCmdShow)            // Window Show State
{
  MSG       msg;                                 // Windows Message Structure
        BOOL  done=FALSE;                        // Bool Variable To Exit Loop

        // Ask The User Which Screen Mode They Prefer
        if (MessageBox(NULL,"Would You Like To Run In Fullscreen Mode?",
            "Start FullScreen?",MB_YESNO|MB_ICONQUESTION)==IDNO)
        {
            fullscreen=FALSE;                    // Windowed Mode
        }

        // Create Our OpenGL Window
        if (!CreateGLWindow(" Particle Engine",800,600,16,fullscreen))
        {
            return 0;               // Quit If Window Was Not Created
        }
        while(!done)                // Loop That Runs While done=FALSE
        {
            if (PeekMessage(&msg,NULL,0,0,PM_REMOVE))  // Is There A Message Waiting?
            {
                if (msg.message==WM_QUIT)  // Have We Received A Quit Message?
                {
                    done=TRUE;             // If So done=TRUE
                }
                else                       // If Not, Deal With Window Messages
                {
                    TranslateMessage(&msg); // Translate The Message
                    DispatchMessage(&msg); // Dispatch The Message
                }
            }
            else                           // If There Are No Messages
            {
              // Draw The Scene.  Watch For ESC Key And Quit Messages From DrawGLScene()
```

```
                    // Active?  Was There A Quit Received?
                    if ((active && !DrawGLScene()) || keys[VK_ESCAPE])
                    {
          done=TRUE;        // ESC or DrawGLScene Signalled A Quit
                    }
                    else      // Not Time To Quit, Update Screen
                    {
                      SwapBuffers(hDC);     // Swap Buffers (Double Buffering)

                    }
               }
        }
        // Shutdown
        KillGLWindow();                     // Kill The Window
        return (msg.wParam);                // Exit The Program
}
```

برنامه **BC++ Builder** فصل :

تنظیمات BC++ Builder آنچنان تفاوتی با VC++ ندارد ، به همین جهت در ادامه برنامه ای ساده به این زبان ارائه می گردد .

```
//fGL.cpp file
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "fGL.h"
#include <gl/gl.h>
#include <gl/glu.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TGLV *GLV;
//---------------------------------------------------------------------------
__fastcall TGLV::TGLV(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TGLV::FormCreate(TObject *Sender)
{
  int pf;

  hDC = GetDC(Handle);

  memset(&pfd, 0, 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   = 32;

  pf = ChoosePixelFormat(hDC, &pfd);
  if (pf == 0) { MessageBox(NULL,
      "ChoosePixelFormat() failed: Cannot find a suitable pixel format.", "Error", MB_OK); return; }
  if (SetPixelFormat(hDC, pf, &pfd) == FALSE) {
    MessageBox(NULL, "SetPixelFormat() failed: Cannot set format specified.", "Error", MB_OK);
    return;
   }
  DescribePixelFormat(hDC, pf, sizeof(PIXELFORMATDESCRIPTOR), &pfd);

  hRC = wglCreateContext(hDC);
  wglMakeCurrent(hDC, hRC);
  ReleaseDC(hDC, Handle);

  GLInit();
}
//---------------------------------------------------------------------------
void TGLV::GLInit()
{
  //LIGHTING
  float ambience[] = { 0.75, 0.75, 0.75, 1.0};
  float lightpos[] = {3.0, 3.0, 3.0, 1.0 };
  glMaterialf(GL_FRONT, GL_SHININESS, 75.0);
  glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambience);
  glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);

  //DRAWING
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClearDepth(100.0);
  glDepthFunc(GL_LESS);
  glEnable(GL_DEPTH_TEST);
  glCullFace(GL_BACK);
  glEnable(GL_CULL_FACE);
  glShadeModel(GL_SMOOTH);
}
//---------------------------------------------------------------------------
void __fastcall TGLV::FormResize(TObject *Sender)
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(20.0f, (float)ClientWidth/(float)ClientHeight, 0.1f, 100.0f);
  glMatrixMode(GL_MODELVIEW);
  glViewport(0.0, 0.0, ClientWidth, ClientHeight);
}
//---------------------------------------------------------------------------
void __fastcall TGLV::FormPaint(TObject *Sender)
{
  DrawScene();
}
//---------------------------------------------------------------------------
void TGLV::DrawScene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
  glLoadIdentity();
  gluLookAt(0,0,3,  5,5,1,  0,0,1);

  glPushMatrix();
    glTranslatef(5,5,0);
    Box();
  glPopMatrix();

  glFinish();
  SwapBuffers(wglGetCurrentDC());
}
//----------------------------------------------------------------------------
void TGLV::Box()
{
  float clr[] = {1,0,0,1};
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, clr);
  glColor4fv(clr);

  glBegin(GL_QUADS);
    glNormal3f(1,0,0);
    glVertex3f(1,0,0);
    glVertex3f(1,1,0);
    glVertex3f(1,1,1);
    glVertex3f(1,0,1);

    glNormal3f(-1,0,0);
    glVertex3f(0,0,0);
    glVertex3f(0,0,1);
    glVertex3f(0,1,1);
    glVertex3f(0,1,0);

    glNormal3f(0,1,0);
    glVertex3f(0,0,0);
    glVertex3f(1,0,0);
    glVertex3f(1,0,1);
    glVertex3f(0,0,1);

    glNormal3f(0,-1,0);
    glVertex3f(0,1,0);
    glVertex3f(0,1,1);
    glVertex3f(1,1,1);
    glVertex3f(1,1,0);

    glNormal3f(0,0,1);
    glVertex3f(0,0,1);
    glVertex3f(1,0,1);
    glVertex3f(1,1,1);
    glVertex3f(0,1,1);

    glNormal3f(0,0,-1);
    glVertex3f(0,0,0);
    glVertex3f(0,1,0);
    glVertex3f(1,1,0);
    glVertex3f(1,0,0);
  glEnd();
}
//----------------------------------------------------------------------------
```

فایل fGL.h:

```
//---------------------------------------------------------------------------
#ifndef fGLH
#define fGLH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TGLV : public TForm
{
__published:    // IDE-managed Components
  void __fastcall FormPaint(TObject *Sender);
  void __fastcall FormResize(TObject *Sender);
  void __fastcall FormCreate(TObject *Sender);
private: // User declarations
public:          // User declarations
  __fastcall TGLV(TComponent* Owner);

    HDC   hDC;
    HGLRC hRC;
    PIXELFORMATDESCRIPTOR pfd;
    void TGLV::GLInit();
    void TGLV::DrawScene();
    void TGLV::Box();
};
//---------------------------------------------------------------------------
extern PACKAGE TGLV *GLV;
//---------------------------------------------------------------------------
#endif
```

فایل pGL.cpp:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("pGL.res");
USEFORM("fGL.cpp", GLV);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
    Application->Initialize();
    Application->CreateForm(__classid(TGLV), &GLV);
    Application->Run();
  }
  catch (Exception &exception)
  {
    Application->ShowException(&exception);
  }
  return 0;
}
//---------------------------------------------------------------------------
```

استفاده از OpenGL در زبان اسمبلی ۳۲ بیتی :

```
; ------------------------------------------
; Title: OpenGL test
;
;
; Notes: Just a little example on how to use OpenGL.
;        Needs OpenGl32.dll and Glu32.dll (should be no problem).
;        Object have 2112 faces / 3 (weird) moving light sources
;
; ------------------------------------------

; --------------- Assembler directives
                        .386
                        .model  flat,stdcall
                        option  casemap:none

; --------------- External includes
                        include  \masm32\include\windows.inc
                        include \masm32\include\user32.inc
                        include \masm32\include\kernel32.inc
                        include \masm32\include\gdi32.inc
                        include \masm32\include\opengl32.inc
                        include \masm32\include\glu32.inc

                        includelib \masm32\lib\user32.lib
                        includelib \masm32\lib\kernel32.lib
                        includelib \masm32\lib\gdi32.lib
                        includelib \masm32\lib\gdi32.lib
                        includelib \masm32\lib\opengl32.lib
                        includelib \masm32\lib\glu32.lib

; --------------- Macros Section
szText                  MACRO Name,Text:VARARG
                        LOCAL  lbl
                        jmp    lbl
Name                    db     Text,0
lbl:
                        ENDM

m2m                     MACRO M1, M2
                        push   M2
                        pop    M1
                        ENDM

return                  MACRO arg
                        mov    eax,arg
                        ret
                        ENDM

; --------------- These constants are not defined in windows.inc
PFD_MAIN_PLANE              equ    0
PFD_TYPE_COLORINDEX         equ    1
PFD_TYPE_RGBA               equ    0
```

```
PFD_DOUBLEBUFFER    equ     1
PFD_DRAW_TO_WINDOW  equ     4
PFD_SUPPORT_OPENGL  equ     020h

; ---------- Data Section
                    .data
szDisplayName       db      "TestOpenGL ",0
                    even
PixFrm              PIXELFORMATDESCRIPTOR <>

CommandLine         dd      0
hWnd                dd      0
MainHDC             dd      0
OpenDC              dd      0
hInstance           dd      0

; Some values
Value0Flt           dd      0.0
Value1Flt           dd      1.0
Value1Dbl           dq      1.0
Value45Dbl          dq      45.0
Value3Dbl           dq      3.0
Value7Dbl           dq      7.0

; Light position
LightSourcePosition     dd      -2.0,-2.0,-4.0,0.0
LightSource2Position    dd      2.0,2.0,4.0,0.0
LightSource3Position    dd      -2.0,2.0,4.0,0.0
LightAmbient        dd      0.2,0.0,0.0,1.0
Light2Ambient       dd      0.0,0.2,0.0,1.0
Light3Ambient       dd      0.0,0.0,0.2,1.0
LightDiffuse        dd      1.0,1.0,1.0,1.0
LightSpecular       dd      1.0,1.0,1.0,1.0
SpotCut             dd      -1.0
SpotExp             dd      0.0
SpotDir             dd      -1.0,-1.0,-1.0
LightConstAtt       dd      1.0
LightLinAtt         dd      1.0
LightQuadAtt        dd      1.0

; --- Sphere 1
; Angles
Sphere1AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere1AnglesSpeedFlt   dd      -1.0,-1.2,-1.4
; Objects datas
Sphere1Color        dd      0.6,0.4,0.2,0.0
Sphere1Radius       dq      1.0
sphere1Parts        dd      24
Sphere1Position     dd      0.0,0.0,-5.0
GlSphere1           dd      0

; --- Sphere 2
; Angles
Sphere2AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere2AnglesSpeedFlt   dd      -1.0,0.8,-0.4
; Objects datas
```

```
Sphere2Color            dd      0.4,0.6,0.2,0.0
Sphere2Radius           dq      0.3
sphere2Parts            dd      16
Sphere2Position         dd      0.0,0.0,-1.5
GlSphere2               dd      0

; --- Sphere 3
; Angles
Sphere3AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere3AnglesSpeedFlt   dd      -1.0,-0.8,1.4
; Objects datas
Sphere3Color            dd      0.2,0.4,0.6,0.0
Sphere3Radius           dq      0.3
sphere3Parts            dd      16
Sphere3Position         dd      0.0,0.0,1.5
GlSphere3               dd      0

; --- Sphere 4
; Angles
Sphere4AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere4AnglesSpeedFlt   dd      -1.0,1.4,-0.8
; Objects datas
Sphere4Color            dd      0.4,0.2,0.6,0.0
Sphere4Radius           dq      0.3
sphere4Parts            dd      16
Sphere4Position         dd      0.0,1.5,0.0
GlSphere4               dd      0

; --- Sphere 5
; Angles
Sphere5AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere5AnglesSpeedFlt   dd      1.0,1.8,0.8
; Objects datas
Sphere5Color            dd      0.6,0.2,0.4,0.0
Sphere5Radius           dq      0.3
sphere5Parts            dd      16
Sphere5Position         dd      1.5,0.0,0.0
GlSphere5               dd      0

; --- Sphere 6
; Angles
Sphere6AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
Sphere6AnglesSpeedFlt   dd      1.0,-1.8,2.0
; Objects datas
Sphere6Color            dd      0.2,0.6,0.4,0.0
Sphere6Radius           dq      0.3
sphere6Parts            dd      16
Sphere6Position         dd      -1.5,0.0,0.0
GlSphere6               dd      0

; --- Sphere 7
; Angles
Sphere7AnglesFlt        dd      0.0,0.0,0.0
; Rotations speed
```

```
Sphere7AnglesSpeedFlt dd      -2.1,-1.8,2.0
; Objects datas
Sphere7Color          dd      0.6,0.6,0.6,0.0
Sphere7Radius         dq      0.3
sphere7Parts          dd      16
Sphere7Position       dd      0.0,-1.5,0.0
GlSphere7             dd      0

; ---------- Procedures Declarations
                      .code

MainInit        PROTO :DWORD,:DWORD,:DWORD,:DWORD
MainLoop              PROTO :DWORD,:DWORD,:DWORD,:DWORD
TopXY                 PROTO :DWORD,:DWORD
DrawScene             PROTO
GlInit                PROTO :DWORD,:DWORD
ResizeObject          PROTO :DWORD,:DWORD
CreateSphere          PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
SetLightSource        PROTO :DWORD,:DWORD,:DWORD
RotateObject          PROTO :DWORD,:DWORD,:DWORD,:DWORD
DeleteSpheres         PROTO

; ---------- Procedures Section
CenterForm            PROC  wDim:DWORD, sDim:DWORD
                      shr   sDim,1
                      shr   wDim,1
                      mov   eax,wDim
                      sub   sDim,eax
                      return sDim
CenterForm            ENDP

DoEvents              PROC
                      LOCAL msg:MSG
StartLoop:            ; Check for waiting messages
                      invoke PeekMessage,ADDR msg,0,0,0,PM_NOREMOVE
                      or    eax,eax
                      jz    NoMsg
                      invoke GetMessage,ADDR msg,NULL,0,0
                      or    eax,eax
                      jz    ExitLoop
                      invoke TranslateMessage,ADDR msg
                      invoke DispatchMessage,ADDR msg
                      jmp   StartLoop
NoMsg:                ; No pending messages: draw the scene
                      invoke DrawScene
                      jmp   StartLoop
ExitLoop:             mov   eax,msg.wParam
                      ret
DoEvents              ENDP

; ---------- Program start
start:                invoke GetModuleHandle,NULL
                      mov   hInstance, eax
                      invoke GetCommandLine
                      mov   CommandLine,eax
                      invoke MainInit,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
                      invoke ExitProcess,eax
```

```
; ---------- Program main inits
MainInit                PROC
        hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,CmdShow:DWORD
                        LOCAL  wc:WNDCLASSEX
                        LOCAL  Wwd:DWORD
                        LOCAL  Wht:DWORD
                        LOCAL  Wtx:DWORD
                        LOCAL  Wty:DWORD
                szText  szClassName,"Win32SDI_Class"
                        mov    wc.cbSize,sizeof WNDCLASSEX
                        mov    wc.style,0
                        mov    wc.lpfnWndProc,offset MainLoop
                        mov    wc.cbClsExtra,NULL
                        mov    wc.cbWndExtra,NULL
                        m2m    wc.hInstance,hInst
                        mov    wc.hbrBackground,COLOR_WINDOWTEXT+1
                        mov    wc.lpszMenuName,NULL
                        mov    wc.lpszClassName,offset szClassName
                        invoke LoadIcon,hInst,2
                        mov    wc.hIcon,eax
                        invoke LoadCursor,NULL,IDC_ARROW
                        mov    wc.hCursor,eax
                        mov    wc.hIconSm,0
                        invoke RegisterClassEx, ADDR wc
                        mov    Wwd,400
                        mov    Wht,420
                        invoke GetSystemMetrics,SM_CXSCREEN
                        invoke CenterForm,Wwd,eax
                        mov    Wtx,eax
                        invoke GetSystemMetrics,SM_CYSCREEN
                        invoke CenterForm,Wht,eax
                        mov    Wty,eax
                        invoke CreateWindowEx,0,ADDR szClassName,
                                              ADDR szDisplayName,
                        WS_OVERLAPPEDWINDOW or WS_CLIPSIBLINGS or WS_CLIPCHILDREN,
                                              Wtx,Wty,Wwd,Wht,
                                              NULL,NULL,
                                              hInst,NULL
                        mov    hWnd,eax
                        invoke LoadMenu,hInst,600
                        invoke SetMenu,hWnd,eax
                        invoke ShowWindow,hWnd,SW_SHOWNORMAL
                        invoke UpdateWindow,hWnd
                        call   DoEvents
                        ret
MainInit                ENDP

; ---------- Program main loop
MainLoop                PROC   hWin:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
                        LOCAL  WINRect:RECT
                        LOCAL  PixFormat:DWORD
                        .if uMsg == WM_COMMAND
                                .if wParam == 1000
                        invoke  SendMessage,hWin,WM_SYSCOMMAND,SC_CLOSE,NULL
                                .endif
                                return  0
                        .elseif  uMsg == WM_CREATE
                                invoke  GetDC,hWin
```

```
                        mov     MainHDC,eax
                        mov     ax,SIZEOF PixFrm
                        mov     PixFrm.nSize,ax
                        mov     PixFrm.nVersion,1
                        mov     PixFrm.dwFlags,PFD_DRAW_TO_WINDOW or
PFD_SUPPORT_OPENGL or PFD_DOUBLEBUFFER
                        mov     PixFrm.dwLayerMask,PFD_MAIN_PLANE
                        mov     PixFrm.iPixelType,PFD_TYPE_RGBA
                        mov     PixFrm.cColorBits,8
                        mov     PixFrm.cDepthBits,16
                        mov     PixFrm.cAccumBits,0
                        mov     PixFrm.cStencilBits,0
                        invoke  ChoosePixelFormat,MainHDC,ADDR PixFrm
                        mov     PixFormat,eax
                        invoke  SetPixelFormat,MainHDC,PixFormat,ADDR PixFrm
                        or      eax,eax
                        jz      NoPixelFmt
                        invoke  wglCreateContext,MainHDC
                        mov     OpenDC,eax
                        invoke  wglMakeCurrent,MainHDC,OpenDC
                        invoke  GetClientRect,hWin,ADDR WINRect
                        invoke  GlInit,WINRect.right,WINRect.bottom
NoPixelFmt:
                        return  0
                .elseif uMsg == WM_SIZE
                        invoke  GetClientRect,hWin,ADDR WINRect
                        invoke  ResizeObject,WINRect.right,WINRect.bottom
                        return  0

                .elseif uMsg == WM_CLOSE
                        szText TheText,"Are you sure ?"
                        invoke  MessageBox,hWin,ADDR TheText,ADDR
szDisplayName,MB_YESNO+MB_ICONQUESTION
                        .if eax == IDNO
                                return  0
                        .endif
                        mov     eax,OpenDC
                        or      eax,eax
                        jz      NoGlDC
                        ; Delete our objects
                        invoke  DeleteSpheres
                        invoke  wglDeleteContext,OpenDC
NoGlDC:                         invoke  ReleaseDC,hWin,MainHDC
                        invoke  DestroyWindow,hWin
                        return  0
                .elseif uMsg == WM_DESTROY
                        invoke  PostQuitMessage,NULL
                        return  0
                .endif
                invoke  DefWindowProc,hWin,uMsg,wParam,lParam
                ret
MainLoop        ENDP

; ------------------------------
; OpenGl related stuff
; ------------------------------

; --------------- Init the scene
```

```
GlInit                  PROC    ParentW:DWORD,ParentH:DWORD
                        invoke  SetLightSource,GL_LIGHT0,ADDR LightSourcePosition,ADDR
LightAmbient
                        invoke  SetLightSource,GL_LIGHT1,ADDR LightSource2Position,ADDR
Light2Ambient
                        invoke  SetLightSource,GL_LIGHT2,ADDR LightSource3Position,ADDR
Light3Ambient
                        invoke  CreateSphere,1,GLU_FILL,GLU_SMOOTH,ADDR Sphere1Color,ADDR
Sphere1Radius,sphere1Parts
                        mov     GlSphere1,eax
                        invoke  CreateSphere,2,GLU_FILL,GLU_SMOOTH,ADDR Sphere2Color,ADDR
Sphere2Radius,sphere2Parts
                        mov     GlSphere2,eax
                        invoke  CreateSphere,3,GLU_FILL,GLU_SMOOTH,ADDR Sphere3Color,ADDR
Sphere3Radius,sphere3Parts
                        mov     GlSphere3,eax
                        invoke  CreateSphere,4,GLU_FILL,GLU_SMOOTH,ADDR Sphere4Color,ADDR
Sphere4Radius,sphere4Parts
                        mov     GlSphere4,eax
                        invoke  CreateSphere,5,GLU_FILL,GLU_SMOOTH,ADDR Sphere5Color,ADDR
Sphere5Radius,sphere5Parts
                        mov     GlSphere5,eax
                        invoke  CreateSphere,6,GLU_FILL,GLU_SMOOTH,ADDR Sphere6Color,ADDR
Sphere6Radius,sphere6Parts
                        mov     GlSphere6,eax
                        invoke  CreateSphere,7,GLU_FILL,GLU_SMOOTH,ADDR Sphere7Color,ADDR
Sphere7Radius,sphere7Parts
                        mov     GlSphere7,eax
                        ; Set global flags
                        invoke  glEnable,GL_DEPTH_TEST
                        invoke  glEnable,GL_LIGHTING
                        invoke  glEnable,GL_CULL_FACE          ; Don't render back faces
                        invoke  glShadeModel,GL_SMOOTH
                        invoke  glEnable,GL_NORMALIZE
                        ret
GlInit                  ENDP

; ---------- Display the scene
DrawScene               PROC
                        invoke  glClear,GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT
                        invoke  glPushMatrix
                                invoke  RotateObject,1,ADDR Sphere1Position,ADDR
Sphere1AnglesSpeedFlt,ADDR Sphere1AnglesFlt
                                invoke  glPushMatrix
                                        invoke  RotateObject,2,ADDR Sphere2Position,ADDR
Sphere2AnglesSpeedFlt,ADDR Sphere2AnglesFlt
                                        invoke  glLightfv,GL_LIGHT0,GL_POSITION,ADDR
LightSourcePosition
                                invoke  glPopMatrix
                                invoke  glPushMatrix
                                        invoke  RotateObject,3,ADDR Sphere3Position,ADDR
Sphere3AnglesSpeedFlt,ADDR Sphere3AnglesFlt
                                        invoke  glLightfv,GL_LIGHT1,GL_POSITION,ADDR
LightSource2Position
                                invoke  glPopMatrix
                                invoke  glPushMatrix
                                        invoke  RotateObject,4,ADDR Sphere4Position,ADDR
Sphere4AnglesSpeedFlt,ADDR Sphere4AnglesFlt
```

```
                                invoke  glLightfv,GL_LIGHT2,GL_POSITION,ADDR
LightSource3Position
                        invoke  glPopMatrix
                        invoke  glPushMatrix
                                invoke  RotateObject,5,ADDR Sphere5Position,ADDR
Sphere5AnglesSpeedFlt,ADDR Sphere5AnglesFlt
                        invoke  glPopMatrix
                        invoke  glPushMatrix
                                invoke  RotateObject,6,ADDR Sphere6Position,ADDR
Sphere6AnglesSpeedFlt,ADDR Sphere6AnglesFlt
                        invoke  glPopMatrix
                        invoke  glPushMatrix
                                invoke  RotateObject,7,ADDR Sphere7Position,ADDR
Sphere7AnglesSpeedFlt,ADDR Sphere7AnglesFlt
                        invoke  glPopMatrix
                invoke  glPopMatrix
                invoke  SwapBuffers,MainHDC
                ret
DrawScene       ENDP

; --------------- Resize the scene
ResizeObject    PROC    ParentW:DWORD,ParentH:DWORD
                invoke  glViewport,0,0,ParentW,ParentH
                invoke  glMatrixMode,GL_PROJECTION
                invoke  glLoadIdentity
                invoke  gluPerspective,DWORD PTR Value45Dbl,DWORD PTR
Value45Dbl+4,DWORD PTR Value1Dbl,DWORD PTR Value1Dbl+4,DWORD PTR Value3Dbl,DWORD
PTR Value3Dbl+4,DWORD PTR Value7Dbl,DWORD PTR Value7Dbl+4
                invoke  glMatrixMode,GL_MODELVIEW
                invoke  glLoadIdentity
                ret
ResizeObject    ENDP

; --------------- Free the memory allocated for the spheres
DeleteSpheres   PROC
                invoke  gluDeleteQuadric,GlSphere1
                invoke  gluDeleteQuadric,GlSphere2
                invoke  gluDeleteQuadric,GlSphere3
                invoke  gluDeleteQuadric,GlSphere4
                invoke  gluDeleteQuadric,GlSphere5
                invoke  gluDeleteQuadric,GlSphere6
                invoke  gluDeleteQuadric,GlSphere7
                ret
DeleteSpheres   ENDP

; --------------- Create a sphere with glu object
CreateSphere
        PROCListNumber:DWORD,FillType:DWORD,NormalsType:DWORD,Color:DWORD,Radius:
        DWORD,Parts:DWORD
                LOCAL   GlSphere:DWORD
                invoke  glNewList,ListNumber,GL_COMPILE
                ; Create a template
                invoke  gluNewQuadric
                mov     GlSphere,eax
                ; Set object draw style
                invoke  gluQuadricDrawStyle,GlSphere,FillType
                ; Set normals style
                invoke  gluQuadricNormals,GlSphere,NormalsType
```

```
                    ; Set object color
                    invoke  glMaterialfv,GL_FRONT,GL_AMBIENT_AND_DIFFUSE,Color
                    mov     eax,Radius
                    ; Create a sphere primitive
                    invoke  gluSphere,GlSphere,[eax],[eax+4],Parts,Parts
                    invoke  glEndList
                    mov     eax,GlSphere
                    ret
CreateSphere        ENDP

; ---------- Set a light source
SetLightSource      PROC    LtNumber:DWORD,LtPosition:DWORD,LtAmbient:DWORD
                    invoke  glLightfv,LtNumber,GL_POSITION,LtPosition
                    invoke  glLightfv,LtNumber,GL_DIFFUSE,ADDR LightDiffuse
                    invoke  glLightfv,LtNumber,GL_SPECULAR,ADDR LightSpecular
                    invoke  glLightfv,LtNumber,GL_AMBIENT,LtAmbient
                    invoke  glLightfv,LtNumber,GL_CONSTANT_ATTENUATION,ADDR
LightConstAtt
                    invoke  glLightfv,LtNumber,GL_LINEAR_ATTENUATION,ADDR LightLinAtt
                    invoke  glLightfv,LtNumber,GL_QUADRATIC_ATTENUATION,ADDR
LightQuadAtt
                    invoke  glLightfv,LtNumber,GL_SPOT_CUTOFF,ADDR SpotCut
                    invoke  glLightfv,LtNumber,GL_SPOT_EXPONENT,ADDR SpotExp
                    invoke  glLightfv,LtNumber,GL_SPOT_DIRECTION,ADDR SpotDir
                    invoke  glEnable,LtNumber
                    ret
SetLightSource      ENDP

; ---------- Position and rotate an object
RotateObject        PROC
        ListNumber:DWORD,XYZPosition:DWORD,XYZRotations:DWORD,XYZAngles:DWORD
                    LOCAL   LocXSpd:DWORD
                    LOCAL   LocYSpd:DWORD
                    LOCAL   LocZSpd:DWORD
                    LOCAL   LocXAngle:DWORD
                    LOCAL   LocYAngle:DWORD
                    LOCAL   LocZAngle:DWORD
                    mov     eax,XYZPosition
                    mov     ecx,[eax]
                    mov     ebx,[eax+4]
                    mov     eax,[eax+8]
                    invoke  glTranslatef,ecx,ebx,eax
                    mov     eax,XYZAngles               ; Rotate X
                    mov     ebx,XYZRotations
                    fld     DWORD PTR [eax]
                    fadd    DWORD PTR [ebx]
                    fstp    DWORD PTR [eax]
                    invoke  glRotatef,[eax],Value0Flt,Value1Flt,Value0Flt
                    mov     eax,XYZAngles               ; Rotate Y
                    mov     ebx,XYZRotations
                    fld     DWORD PTR [eax+4]
                    fadd    DWORD PTR [ebx+4]
                    fstp    DWORD PTR [eax+4]
                    invoke  glRotatef,[eax+4],Value1Flt,Value0Flt,Value0Flt
                    mov     eax,XYZAngles               ; Rotate Z
                    mov     ebx,XYZRotations
                    fld     DWORD PTR [eax+8]
                    fadd    DWORD PTR [ebx+8]
```

```
                fstp    DWORD PTR [eax+8]
                invoke  glRotatef,DWORD PTR [eax+8],Value0Flt,Value0Flt,Value1Flt
                invoke  glCallList,ListNumber
                ret
RotateObject    ENDP

; ---------- Program end
end start
```