Технології програмування КС
лекція 1 (частина 2)

Проф. Цимбал О.М., кафедра ТАВР, ХНУРЕ, Харків, Україна

Література

- Цимбал О.М. Технології програмування: Visual C++. – Х.:, ХНУРЕ, 2006, 334 с. (Розділ 12).
- Ву, М. OpenGL. Официальное руководство программиста [Текст] / М. Ву, Дж. Рейдер, Т. Девис, Д. Шрайнер. – СПб.: «ДиаСофтЮП», 2002. – 592 с.
- Тихомиров, Ю.В. OpenGL. Программирование трехмерной графики [Текст] / Ю.В. Тихомиров – СПб.: БХВ-Петербург, 2002. – 304 с.
- Тарасов И. Основы OpenGL, Учебное пособие, Примеры / Http://www.Opengl.Org.Ru/books/open_gl/index.Html.
- Евченко А.И. OpenGL и DirectX. Программирование графики. - СПб: Питер, 2006. – 350 с.

OpenGL. Основна інформація

OpenGL є графічною бібліотекою, що надає програмний інтерфейс забезпечення машинної графіки (близько 250 команд), які використовуються для визначення об’єктів і операцій, необхідних під час створення тривимірного графічного програмного забезпечення.

OpenGL розроблено у як низькорівневий апаратно-незалежний інтерфейс, що реалізується рядом різних апаратних платформ. З цієї причини у OpenGL не включені команди для роботи з вікнами або забезпечення введення користувача. Також бібліотека не містить команди високого рівня для опису моделей тривимірних об’єктів. Навпаки, необхідна модель створюється з обмеженого набору геометричних примітивів: точок, ліній, багатокутників.

На базі OpenGL пізніше були створені більш складні бібліотеки:
бібліотека утиліт OpenGL (GLU – OpenGL Utility Library), яка включає широкі можливості моделювання (поверхні другого порядку та сплайни),
бібліотека FSG (Fahrenheit Scene Graph).

Конкуренти OpenGL: DirectX (Direct3D), Quartz (Mac OS), GEGL (Gimp), Blend4Web.

OpenGL. Основні можливості

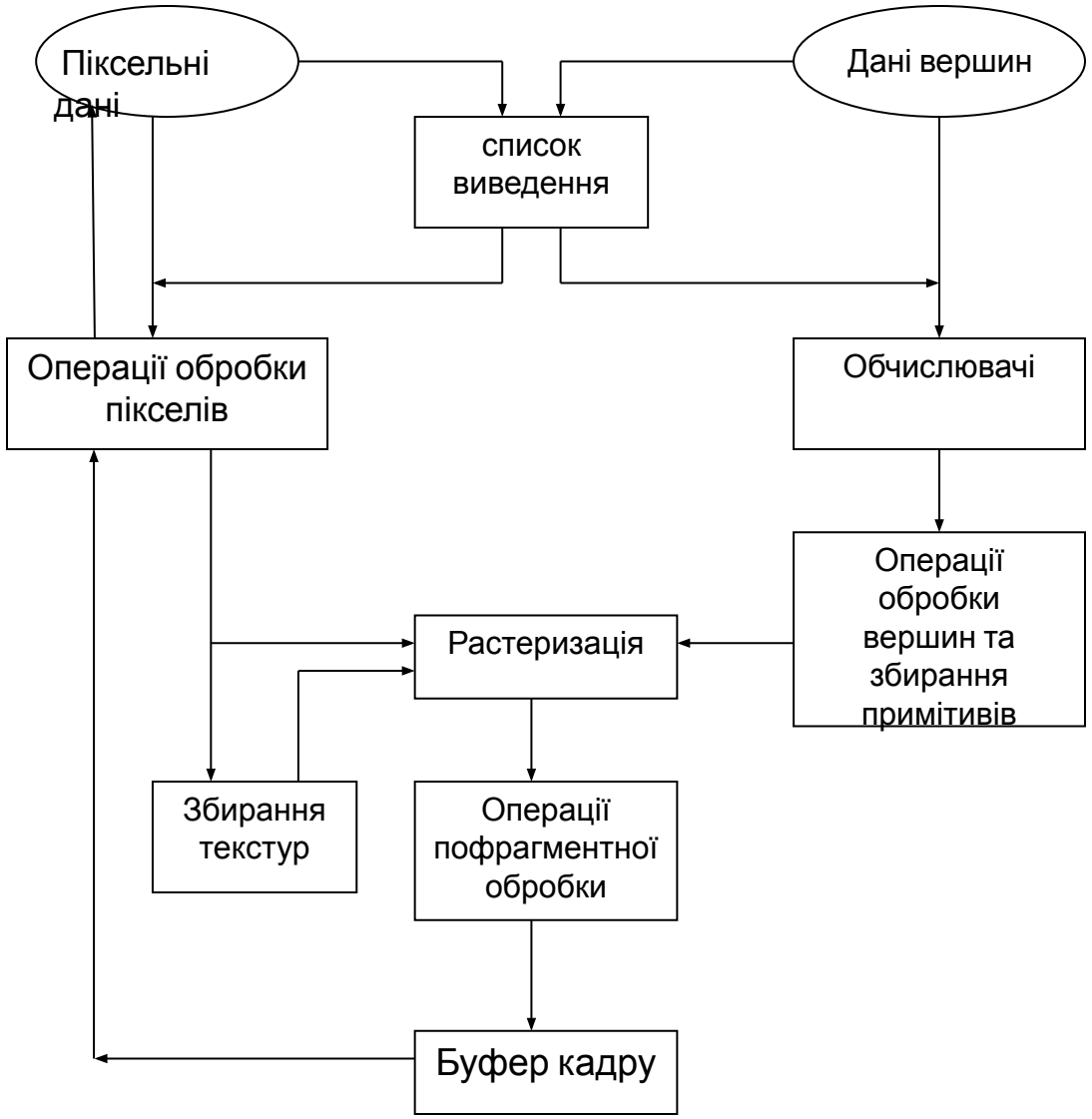
До основних можливості OpenGL:

- відображення каркасних моделей;
- підтримка ілюзії перспективи та глибини простору, атмосферних ефектів (туман, сніг);
- підтримка згладжування вершин сцен;
- підтримка сцен з плоским зафарбовуванням без використанням освітлення;
- підтримка сцен з освітленням та рівномірним зафарбовуванням;
- накладення тіней та текстур;
- моделювання розмитості відображення рухомих об’єктів (motion blur);
- підтримка ефектів глибини різко зображеного простору.

Основними графічними операціями OpenGL є:

- створення форм з графічних примітивів, і таким чином, створення математичних описів об’єктів (примітивами є лінії, багатокутники, зображення);
- упорядкування об’єктів у тривимірному просторі та вибір бажаної точки розташування камери для перегляду скомпонованої сцени;
- обчислення кольорів усіх об’єктів, де кольори у явний вигляд задаються програмою, визначаються, виходячи з умов освітлення, отримуються шляхом накладання текстур або комбінацією трьох зазначених дій;
- перетворення математичного опису об’єктів та пов’язаної з ними інформації щодо кольору об’єктів у пікселі на екрані (процес, що називають растеризацією).

OpenGL. Конвеєр візуалізації



Візуалізацією є процес, під час якого комп’ютер створює зображення з моделей. У свою чергу, моделі або об’єкти створюються з графічних примітивів – точок, ліній та багатокутників, що визначаються їх вершинами.

Кінцеве візуалізоване зображення складається з пікселів, що виводяться на екран. Піксель є найменшим видимим елементом, що може створюватися апаратними засобами відображення.

З програмної точки зору, OpenGL є набором команд, що визначають координати об’єктів, забезпечують встановлення параметрів та їх перетворення. У будь-якому випадку команди OpenGL мають однаковий та визначений порядок виконання операцій обробки, що називається конвеєром візуалізації OpenGL. Цей порядок є практично повною структурою OpenGL

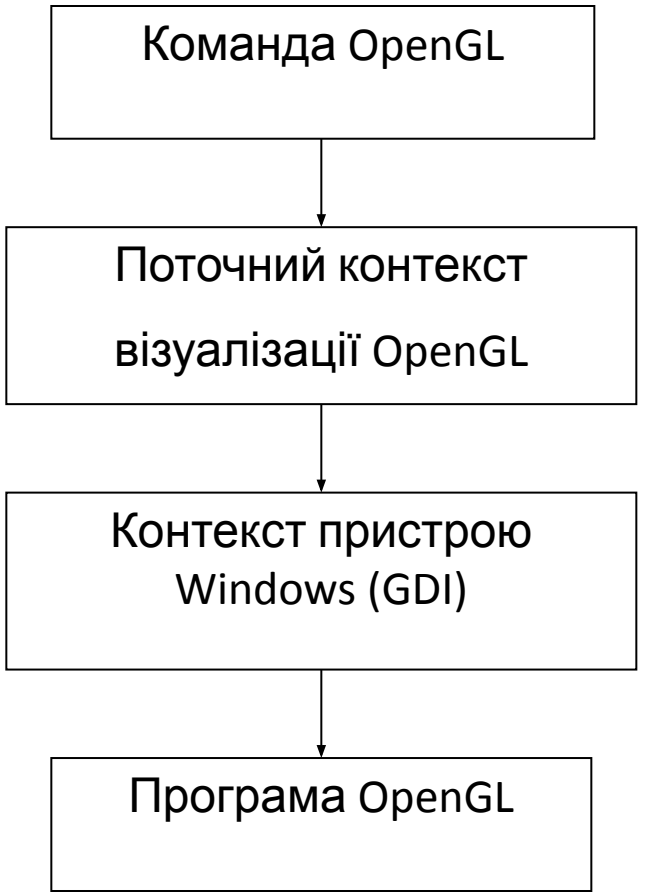
OpenGL. Конвеєр візуалізації

1. Будь-які дані OpenGL можуть зберігатися у списках виведення.
2. Усі графічні примітиви описуються вершинами. Параметричні криві та поверхні можуть спочатку описуватися контрольними точками та поліноміальними (базовими) функціями. Обчислювачі надають методи отримання вершин.
3. На стадії операцій обробки вершин, вершини перетворюються у примітиви. Якщо програма використовує текстури та освітлення, генеруються текстури, обчислюються параметри освітлення,
4. Збирання примітивів включає операція відсікання частин зображення, що виходять за межі визначеного напівпростору. Результатом етапу є закінчені геометричні примітиви.
5. Піксельні дані розпаковуються у компоненти певного формату, потім масштабуються, змішуються, оброблюються за допомогою елементів відображення, фіксуються й записуються в область пам’яті,.
6. На операційїзбирання текстур програми можуть накладати зображення на геометричні об’єкти,
7. Растеризацією є перетворення геометричних або піксельних даних у фрагменти.
8. Перед збереженням у буфері кадру виконується ряд операцій: накладення текстур, обчислення туману, операції змішування кольорів.
9. Буфер кадру відіграє особливу роль. Дані буфера кадру можуть записуватися та зчитуватися функціями OpenGL, однак його настроювання відбувається за допомогою операційної системи, у якій використовується OpenGL.

Взаємодія Windows та OpenGL у MFC-програмах

Особливості:

- програми для виведення інформації графічного або текстового вмісту використовують контексти пристроїв. Інтерфейс графічних пристроїв Windows GDI забезпечує керування контекстами цих пристроїв. Технологія OpenGL не співпрацює зі стандартними контекстами пристроїв Windows, натомість використовує так званий контекст візуалізації (rendering context);
- Під час роботи з контекстом пристрою для вибору нового інструмента відображення (пера або пензля) використовується GDI-функція *SelectObject()*. Обраний інструмент залишається активним до вибору наступного інструмента.
- За аналогією з контекстом пристрою, контекст візуалізації забезпечує обробку графічного стану. У OpenGL має бути присутнім спеціальний механізм взаємозв’язку вихідної інформації з контекстом пристроїв Windows. Шляхом розміщення визначених ділянок інформації у структурі контексту візуалізації, OpenGL дозволяє оновлювати графічний стан вікна під



Взаємодія Windows та OpenGL у MFC-програмах

Контексти візуалізації є багатопотоковими, тобто один спільний контекст візуалізації може використовуватися декількома потоками. У структурі програми може використовуватися декілька контекстів візуалізації.

OpenGL-команди не потребують дескрипторів або покажчиків на контексти візуалізації незалежно від того, який контекст візуалізації є активним. Контекст візуалізації є практично однаковим для різних OpenGL-команд.

У процесі роботи з OpenGL контекст візуалізації спочатку створюється, потім ініціюється, обирається поточним, використовується, відключається та вилучається.

У реальних OpenGL-програмах створений та ініційований контекст візуалізації надалі багаторазово використовується у роботі поточної програми. При цьому активізація та відключення контексту виконуються багаторазово, а вилучення здійснюється один раз – після закінчення роботи програми.

Використання контекстів візуалізації забезпечується такими WinAPI-функціями:

```
HGLRC wglCreateContext(HDC  hdc );
// створення контексту візуалізаціїBOOL
wglDeleteContext(HGLRC  hglrc );       //
вилучення контексту візуалізації
BOOL wglMakeCurrent(HDC hdc,
HGLRC  hglrc );  // активізація контексту візуалізації
```

У наведених функціях *hdc* – дескриптор контексту GDI, *hglrc* – дескриптор контексту візуалізації OpenGL.

Структура PIXELFORMATDESCRIPTOR

Містить інформація щодо взаємодії OpenGL та конкретного системного пристрою Windows.

```
typedef struct tagPIXELFORMATDESCRIPTOR {
WORD  nSize;                    // визначає розмір структури, дорівнює sizeof (PIXELFORMATDESCRIPTOR)
WORD  nVersion;                 // версія структури (має бути 1.0)
DWORD dwFlags;                  //  набір бітів, що характеризують властивості буфера пікселів
BYTE  iPixelType;               // тип піксельних даних
BYTE  cColorBits;               // кількість бітових площин у кожному буфері кольору
BYTE  cRedBits;                 // кількість бітових площин червоного у кожному буфері RGBA
BYTE  cRedShift;                // зсув від початку кількості бітових площин червоного у кожному буфері RGBA
BYTE  cGreenBits;               // кількість бітових площин зеленого у кожному буфері RGBA
BYTE  cGreenShift;              // зсув від початку кількості бітових площин зеленого у кожному буфері RGBA
BYTE  cBlueBits;                // кількість бітових площин синього у кожному буфері RGBA
BYTE  cBlueShift;               // зсув від початку кількості бітових площин синього у кожному буфері RGBA
BYTE  cAlphaBits;               // кількість бітових площин альфа у кожному буфері RGBA
BYTE  cAlphaShift;              // зсув від початку кількості бітових площин альфа у кожному буфері RGBA
BYTE  cAccumBits;          // загальна кількість бітових площин у буфері акумулятора
BYTE  cAccumRedBits;            // загальна кількість бітових площин червоного у буфері акумулятора
BYTE  cAccumGreenBits;          // загальна кількість бітових площин зеленого у буфері акумулятора
BYTE  cAccumBlueBits;           // загальна кількість бітових площин синього у буфері акумулятора
BYTE  cAccumAlphaBits;          // загальна кількість бітових площин альфа у буфері акумулятора
BYTE  cDepthBits;               // розмір буфера глибини (вісь Z)
BYTE  cStencilBits;             // розмір буфера трафарету
BYTE  cAuxBuffers;              // кількість допоміжних буферів (не підтримується)
BYTE  iLayerType;               // ігнорується
BYTE  bReserved;                // кількість площин переднього та заднього плану
DWORD dwLayerMask;              // ігнорується
DWORD dwVisibleMask;            // індекс або колір прозорості нижньої площини
DWORD dwDamageMask;             // ігнорується
} PIXELFORMATDESCRIPTOR
```

Структура простої OpenGL-програми (на базі MFC-проекту)

1. Зміни у класі головного вікна (*.h файл):

```
class CMainWin : public CFrameWnd
{HGLRC m_hrc;
 public:      CMainWin(void);
        void OnSize(unsigned int type, int x, int y);
        void OnPaint(void);
        void GLInit(void);
        void OnOpenGL(void);
        void OnHScroll(UINT SBCode, UINT Pos, CScrollBar *SB);
        void OnVScroll(UINT SBCode, UINT Pos, CScrollBar *SB);
        DECLARE_MESSAGE_MAP()
};
```

2. Зміни у конструкторі головного вікна (*.cpp файл)

```
CMainWin::CMainWin(void)
{Create(NULL,"OpenGL test",WS_OVERLAPPEDWINDOW|
WS_VSCROLL|WS_HSCROLL);
 PIXELFORMATDESCRIPTOR pfd;
 pfd.dwFlags=PFD_DOUBLEBUFFER;
 CClientDC dc(this);
 int nPixelFormat=ChoosePixelFormat(dc.m_hDC,&pfd);
 SetPixelFormat(dc.m_hDC,nPixelFormat,&pfd);
 m_hrc=wglCreateContext(dc.m_hDC);
 wglMakeCurrent(dc.m_hDC,m_hrc);
}
```

3. Обробник повідомлення WM_SIZE (*.cpp файл)

```
void CMainWin::OnSize(unsigned int type, int x, int y)
{CClientDC dc(this);
wglMakeCurrent(dc.m_hDC,m_hrc);
GLdouble gldAspect=(GLdouble)x/(GLdouble)y;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(30.0,gldAspect,1.0,10.0);
glViewport(0,0,x,y);
wglMakeCurrent(NULL,NULL);
}
```

4. Обробник повідомлення WM_PAINT (*.cpp файл)

```
CMainWin::OnPaint(void)
{CPaintDC pDC(this);
 wglMakeCurrent(pDC.m_hDC,m_hrc);
 GLInit();
 OnOpenGL();
 SwapBuffers(pDC.m_hDC);
 wglMakeCurrent(NULL,NULL);
}
```

5. Обробники повідомлень WM_HSCROLL та WM_VSCROLL (див. Лекція 6, семестр 1).

Структура простої OpenGL-програми (на базі MFC-проекту)

6. Функція ініціалізації станів OpenGL (*.cpp файл):

```
void CMainWin::GLInit(void)
{GLdouble marengo[3]={1.0,1.0,0.0};
glClearColor(1.0,1.0,1.0,1.0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0,0.0,-5.0);
glColor3dv(marengo);
glScalef(1.0,1.0,1.0);
}
```

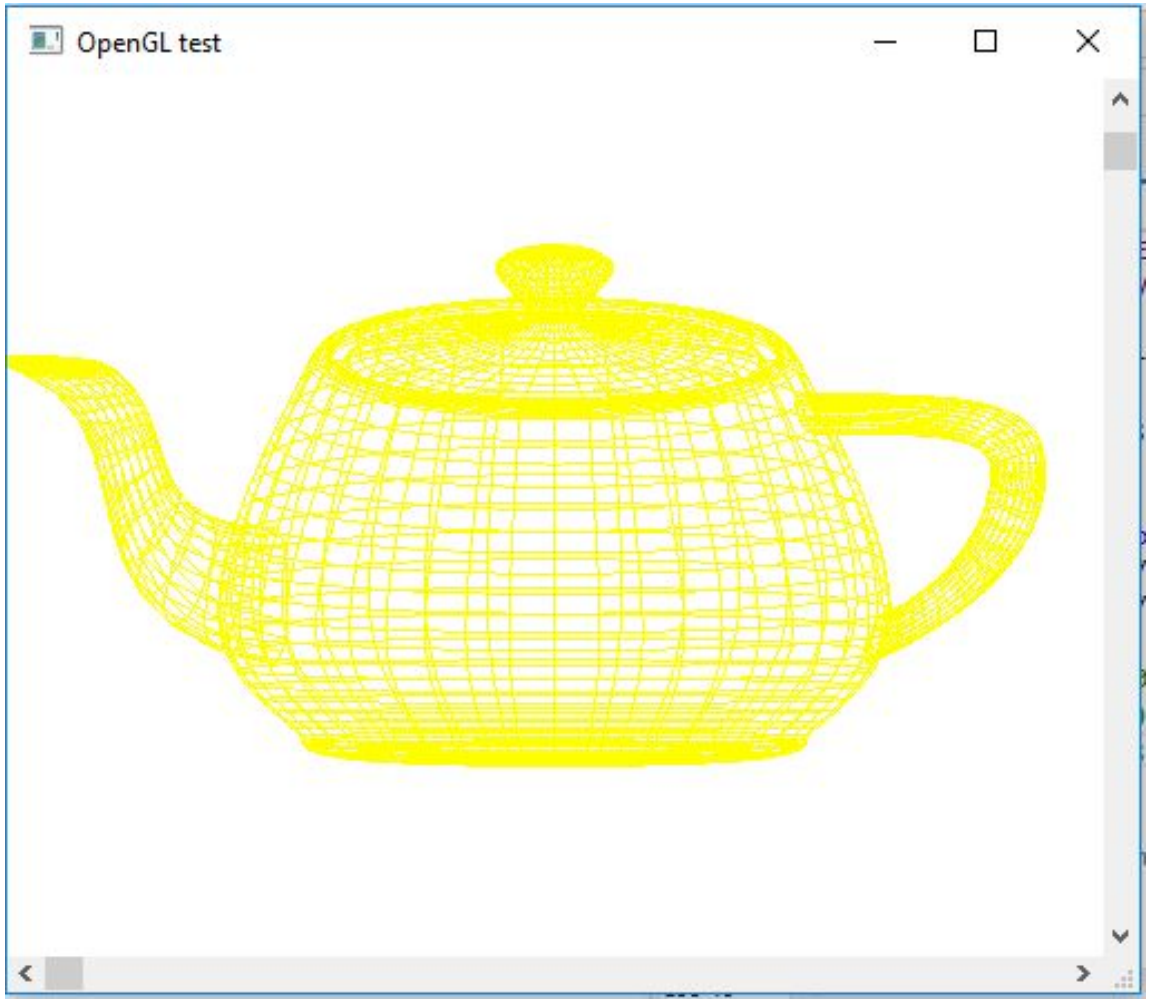
7. Функція виведення графічної сцени (*.cpp файл)

```
void CMainWin::OnOpenGL(void)
{glRotated(360.0*hspos/100,0,1,0);
glRotated(360.0*vspos/100,1,0,0);
auxSolidTeapot(1.0);
}
```

8. Заголовкові файли та глобальні змінні:

```
#include <afxwin.h>
#include "gl/gl.h"
#include "gl/glu.h"
#include "glaux.h“(може не входити до MSVS)
#include "App.h"
int hspos=50,vspos=50;
```

9. Бібліотеки OpenGL , що необхідно додати в проект:
Для Windows 10:
C:\Program Files (x86)\Windows Kits\8.0\Lib\win8\um\x86
glu32.lib, OpenGL32.lib.
glaux.lib (може не входити до MSVS)



Структура простої OpenGL-програми (на базі консольного-проекту)

1. Заголовкові файли, оголошення функцій та змінних

```
#include <windows.h>
#include "gl/gl.h"
#include "gl/glu.h"
#include "glaux.h"
void CALLBACK display(void);
void CALLBACK resize(int, int);
void CALLBACK Key_ESC(void);
void CALLBACK Key_LEFT(void);
void CALLBACK Key_RIGHT(void);
void CALLBACK Key_UP(void);
void CALLBACK Key_DOWN(void);
int angle=0;   bool lr=false,ud=false;
```

2. Функцій main() програми

```
void main()
{   auxInitPosition( 50, 10, 400, 400);
    auxInitDisplayMode( AUX_RGB | AUX_DEPTH | AUX_DOUBLE );
    auxInitWindow(LPCWSTR("Glaux Template"));
    auxIdleFunc(display);   auxReshapeFunc(resize);
    glClearColor(1.0,1.0,1.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    auxKeyFunc(AUX_LEFT, Key_LEFT); auxKeyFunc(AUX_UP, Key_UP);
    auxKeyFunc(AUX_RIGHT, Key_RIGHT);
    auxKeyFunc(AUX_DOWN, Key_DOWN);
    auxMainLoop(display);
}
```

3. Функція resize()

```
void CALLBACK resize(int width,int height)
{  glViewport(0,0,width,height);
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   glOrtho(-5,5, -5,5, 2,12);
   gluLookAt( 0,0,5, 0,0,0, 0,1,0 );
   glMatrixMode( GL_MODELVIEW );
}
```

4. Функція display()

```
void CALLBACK display(void)
{ glClear( GL_COLOR_BUFFER_BIT |
       GL_DEPTH_BUFFER_BIT );
  glColor3d(1,0,0);
  if(lr)glRotatef(angle,1,0,0);
  if(ud)glRotatef(angle,0,1,0);
  auxWireTeapot(2.5);
  auxSwapBuffers();
}
```

Структура простої OpenGL-програми (на базі консольного проекту)

5. Функції-обробники стрілок клавіатури:

```
void CALLBACK Key_ESC(void)  {exit(0); }

void CALLBACK Key_LEFT(void)
{angle+=3;
lr=true;ud=false;
}
void CALLBACK Key_RIGHT(void)
{angle-=3;
lr=true;ud=false;
}

void CALLBACK Key_UP(void)
{angle+=3;
lr=false;ud=true;
}

void CALLBACK Key_DOWN(void)
{angle-=3;
lr=false;ud=true;
}
```