Технології програмування КС
лекція 2 (частина 2)

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

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

1. Заголовкові файли, оголошення функцій та змінних

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

// файли glut32.lib та glut32.dll – в теці проекту

void display(void);
void resize(int,int);
```

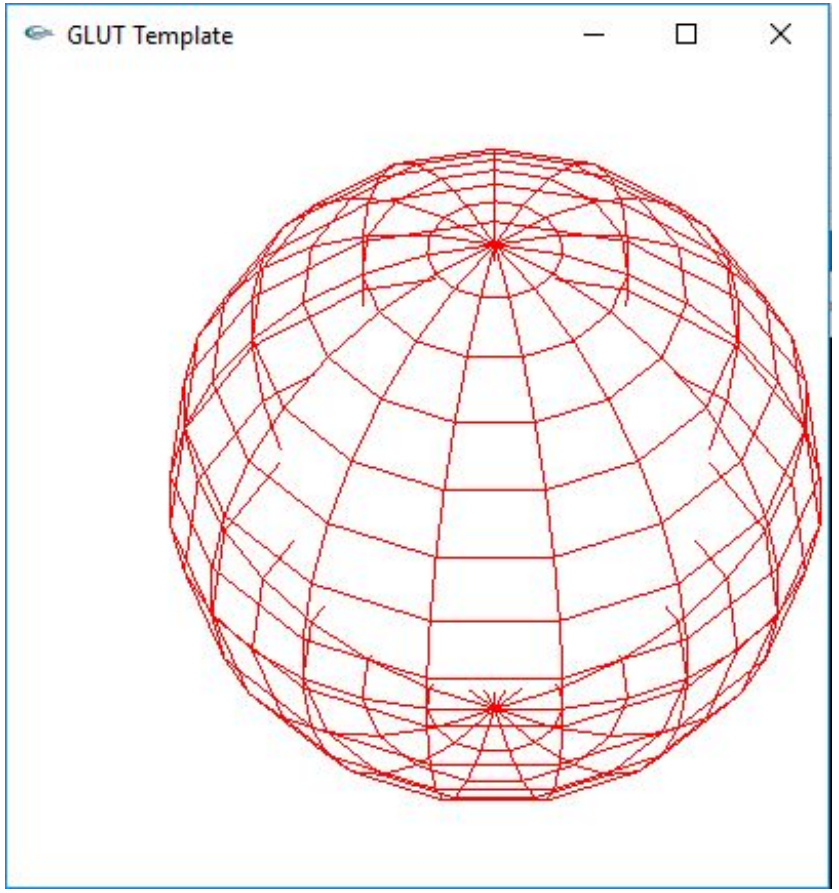
2. Функцій main() програми

```
void main()
{   glutInitWindowPosition(50, 10);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH |
                         GLUT_DOUBLE );
    glutCreateWindow( "GLUT Template" );
    glutIdleFunc(display);
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    /* Enter your cod here */
    glutMainLoop();
}
```

3. Функції resize() та display()

```
void 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);
}

void display(void)
{ glClearColor(1.0,1.0,1.0,1.0);
  glClear(GL_COLOR_BUFFER_BIT|
          GL_DEPTH_BUFFER_BIT);
  glTranslated(0.0005,0,0);
  glRotated(45.0,1.0,0.0,0.0);
  glColor3d(1,0,0);
  glutWireSphere(1, 15,15);
  glutSwapBuffers();
}
```



Синтаксис команд та основні типи OpenGL

Функції та процедури OpenGL, зазвичай, називають командами
Усі команди OpenGL мають префікс *gl*, а кожна назва функції починається з великої літери, наприклад *glColor()*, *glPushMatrix()*.
Узагальнений вигляд команди OpenGL:

rtype CommandName [1 2 3 4] [b s i f d us ui] [v] (atype arg)

де:

CommandName – ім'я OpenGL-команди, наприклад *glVertex*;

[1 2 3 4] – кількість аргументів;

[b s i f d us ui] – тип аргументів;

[v] – вектор, *rtype* – тип що повертається.

atype та *arg* - типом та кількість аргументів.

Варіанти запису команди glVertex:
glVertex2d, glVertex2f, glVertex3d, glVertex3i, glVertex4s, glVertex2dv, glVertex2fv….
Константи OpenGL записують великими літерами, наприклад *GL_COLOR_BUFFER_BIT*.

символ	тип OpenGL	тип C/C++
b	GLbyte	char
s	GLshort	short
i	GLint	int
f	GLfloat	float
d	GLdouble	double
ub	GLbyte	byte
us	GLshort	short
ui	GLuint	uint

Команди відображення графічних примітивів OpenGL

вершини графічних об’єктів OpenGL визначаються командою *glVertex*():*

void glVertex [2 3 4] [s i f d] v (type coord); // приклад – glVertex2i(0, -4);

glVertex()* визначається чотирма координатами: *x*, *y*, *z* та *w*.
glVertex2() задає *x, y, z =0, w=1*. *glVertex3()* задає *x*, *y*, *z, w=1,*
glVertex4() усі 4 координати. *w* враховує перспективу.

Примітив OpenGL – простий графічний об’єкт (точка, лінія, багатокутник, растрове зображення), яким можна маніпулювати як єдиною дискретною сутністю.

Поточний колір об’єкта разом з умовами освітлення визначає сумарний колір, який поширюється на всі пізніше створювані примітиви – до встановлення нового.

Команди кольору:
void glColor [3 4] [b s i f d][v] (GLtype components); // встановлення RGBA-кольору
void glIndex [3 4] [s i f d] v (GLtype index); // індексний режим кольору

glColor3f (0.0, 0.0, 0.0); // black glColor3f (1.0, 0.0, 0.0); // red glColor3f (0.0, 1.0, 0.0); // green

Альфа-параметр визначає ступінь прозорості: 0.0 – відсутня, 1.0 – повна

Команди відображення графічних примітивів OpenGL

За допомогою наборів вершин задаються примітиви або групи примітивів: точки, лінії, з'єднані лінії, замкнені лінії, трикутники різного типу, позв'язані та непозв'язані чотирикутники, полігони.

Формування набору точок забезпечується комбінацією команд *glBegin() / glEnd()*:

```
void glBegin (GLenum mode);// початок послідовності
void glEnd ( );                          // кінець послідовності
```

Всередині комбінації команд *glBegin() / glEnd()* окрім вершин встановлюються: колір, вектор нормалі, координати текстур, властивості матеріалу, виклик списків. Використання інших команд не допускається.

Стандартно точка відображується як окремий піксель, лінія - ширину 1 піксель. Нестандартні значення задаються функціями *glPointSize()* та *glLineWidth()*:

```
void glPointSize   (GLfloat size);    // size – ширина точки у пікселях
void glLineWidth (GLfloat width); // width – ширина лінії у пікселях
```

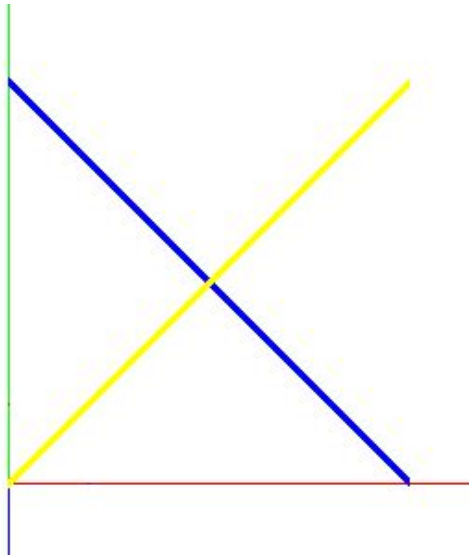
Команди відображення графічних примітивів OpenGL

Значення *mode*	Результат застосування
GL_POINTS	кожна вершина розглядається як окрема незалежна точка
GL_LINES	пара вершин розглядається як незалежний відрізок (перша пара – перший відрізок, друга – другий і т.д.). Остання непарна вершина ігнорується.
GL_LINE_STRIP	послідовність з одного або декількох зв'язаних відрізків. Перша вершина – початок першого відрізка; друга – кінець, одночасно – початок другого відрізка і т.д. (N-1) відрізків.
GL_LINE_LOOP	аналогічний режиму GL_LINE_STRIP. Додатково: остання вершина замикається відрізком на першу. N відрізків.
GL_TRIANGLES	кожна трійка вершин формує незалежний трикутник. Якщо кількість вершин не є кратною 3, решта вершин (1 або 2) ігноруються.
GL_TRIANGLE_STRIP	група зв'язаних трикутників, що мають спільні грані: (1, 2, 3) – перший трикутник; (2, 3, 4) – другий трикутник і т.д. (N-2) трикутників.
GL_TRIANGLE_FAN	група зв'язаних трикутників, що формують веєр із спільною вершиною 1: (1, 2, 3) – перший трикутник; (1, 3, 4)– другий трикутник і т.д. (N-2) трикутників.
GL_QUADS	кожна четвірка вершин формує незалежний чотирикутник: (1, 2, 3, 4) – перший, (5, 6, 7, 8) – другий і т.д. Якщо кількість вершин не є кратною 4, решта вершин (1, 2, 3) ігноруються. N/4 чотирикутників.
GL_QUAD_STRIP	група зв'язаних чотирикутників, що мають спільні грані: (1, 2, 4, 3) – перший, (3, 4, 6, 5) – другий і т.д. Якщо N – непарне число, остання вершина ігнорується.
GL_POLYGON	багатокутник, що будується на послідовності точок (> 3). Багатокутник має не перетинати сам себе і має бути опуклим.

Команди відображення графічних примітивів OpenGL

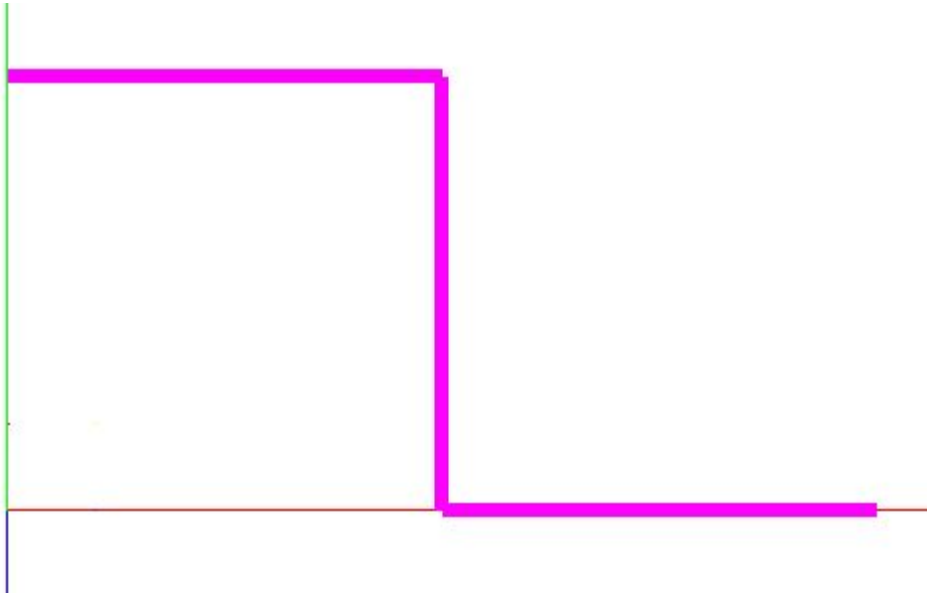
Приклад з незалежними лініями:

```
glLineWidth(5.0);
glBegin(GL_LINES);
        glColor3f(0.0,0.0,1.0);
        glVertex3f(5.0f,0.0f,0.0f); glVertex3f(0.0f,5.0f,0.0f);
        glColor3f(1.0,1.0,0.0);
        glVertex3f(0.0f,0.0f,0.0f); glVertex3f(5.0f,5.0f,0.0f);
glEnd();
```



Приклад з пов’язаними лініями:

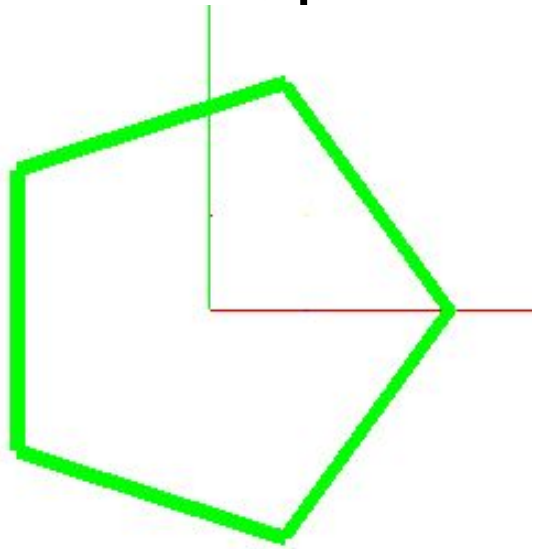
```
glLineWidth(10.0);
glBegin(GL_LINE_STRIP);
glColor3f(1.0,0.0,1.0);
glVertex2i(0,5); glVertex2i(5,5);
glVertex2i(5,0); glVertex2i(10,0);
glEnd();
```



Команди відображення графічних примітивів OpenGL

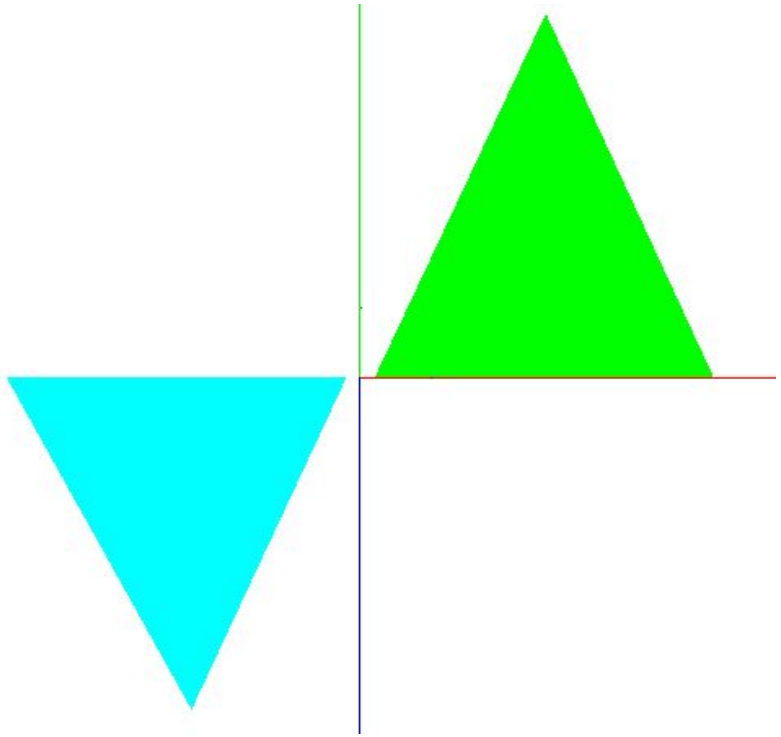
Приклад із замкненою лінією:

```
glLineWidth(15.0);
glBegin(GL_LINE_LOOP);
glColor3f(0.0,1.0,0.0);
      for(int i=0;i<5;i++)glVertex2f(2.5*cos(0.4*pi*i),2.5*sin(0.4*pi*i));
glEnd();
```



Приклад із незалежними трикутниками:

```
glBegin(GL_TRIANGLES);
glColor3f(0.0,1.0,0.0);
glVertex2f(0.2,0.0);  glVertex2f(5.0,0.0); glVertex2f(2.5,5.0);  // 1
glColor3f(0.0,1.0,1.0);
glVertex2f(-0.2,0.0);  glVertex2f(-5.0,0.0); glVertex2f(-2.5,-5.0);  // 2
glEnd();
```



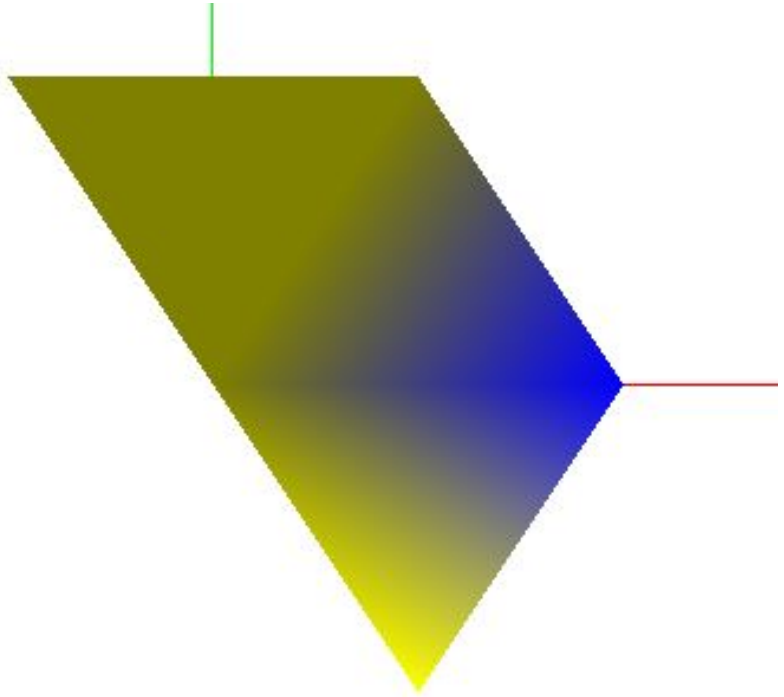
Команди відображення графічних примітивів OpenGL

Приклад із пов’язаними трикутниками:

```
glBegin(GL_TRIANGLE_STRIP);
        glColor3f(1.0,0.5,0.0);
        glVertex2f(0.0,0.0);  glVertex2f(-2.5,-5.0);
        glColor3f(1.0,0.0,1.0);
        glVertex2f(-5.0,0.0); glVertex2f(-7.5,-5.0);
        glColor3f(0.0,1.0,0.0);
        glVertex2f(-10.0,0.0);
glEnd();
```

Приклад із веєром трикутників:

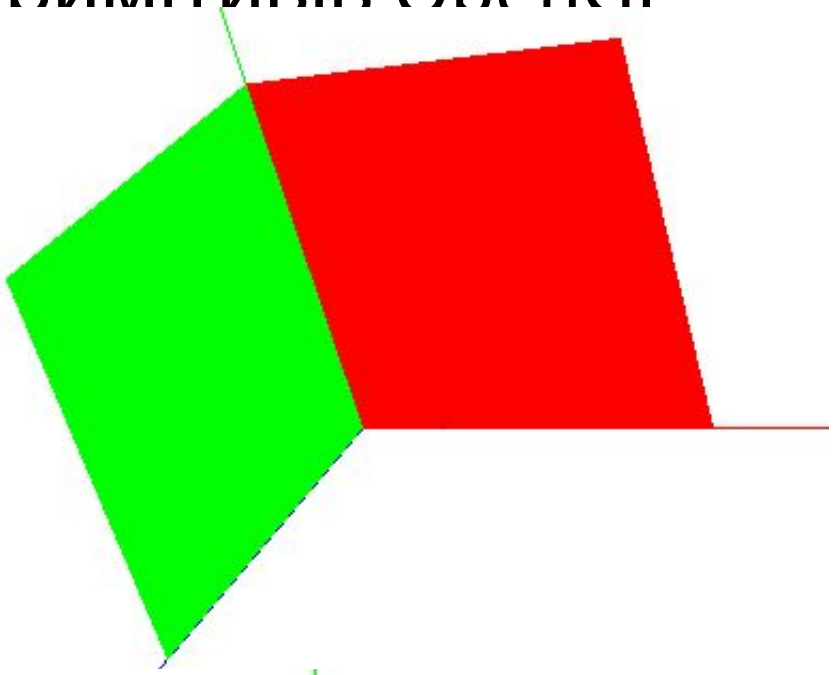
```
glBegin(GL_TRIANGLE_FAN);
        glColor3f(0.5,0.5,0.0);
                glVertex2i(0,0);
                glVertex2i(-2,3);  glVertex2i(2,3);
        glColor3f(0.0,0.0,1.0);
                glVertex2i(4,0);
        glColor3f(1.0,1.0,0.0);
                glVertex2i(2,-3);
glEnd();
```



Команди відображення графічних примітивів OpenGl

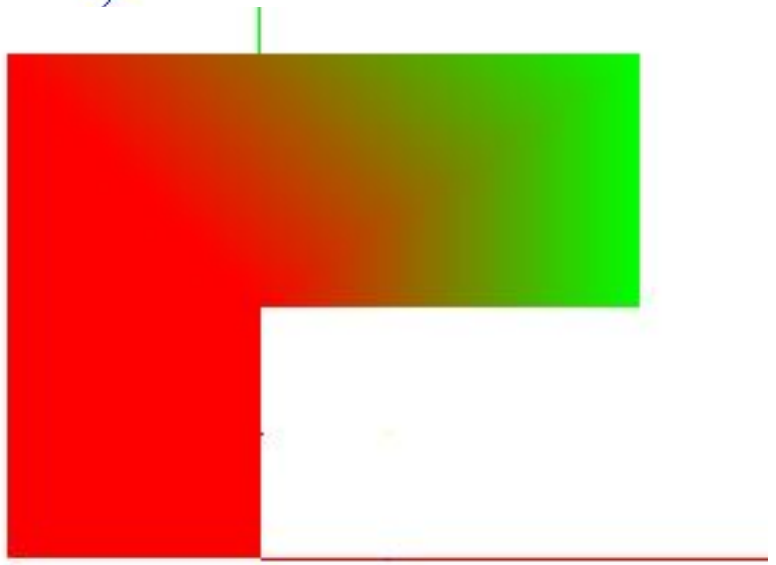
Приклад із незалежними чотирикутниками:

```
glBegin(GL_QUADS);
        glColor3f(1.0,0.0,0.0);
        glVertex2i(0,0); glVertex2i(0,4);  glVertex2i(4,4); glVertex2i(4,0);
        glColor3f(0.0,1.0,0.0);
        glVertex3i(0,0,0); glVertex3i(0,4,0);  glVertex3i(0,4,4); glVertex3i(0,0,4
glEnd();
```



Приклад із пов’язаними чотирикутниками:

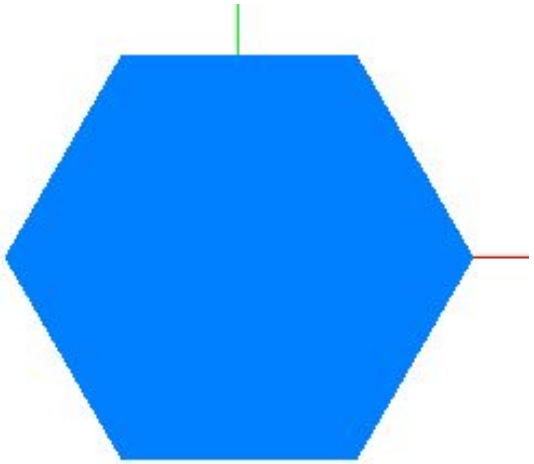
```
glBegin(GL_QUAD_STRIP);
glColor3f(1.0,0.0,0.0);
glVertex2i(0,0); glVertex2i(-2,0);   glVertex2i(0,2);  glVertex2i(-2,4);
glColor3f(0.0,1.0,0.0);
glVertex2i(3,2); glVertex2i(3,4);
glEnd();
```



Команди відображення графічних примітивів OpenGL

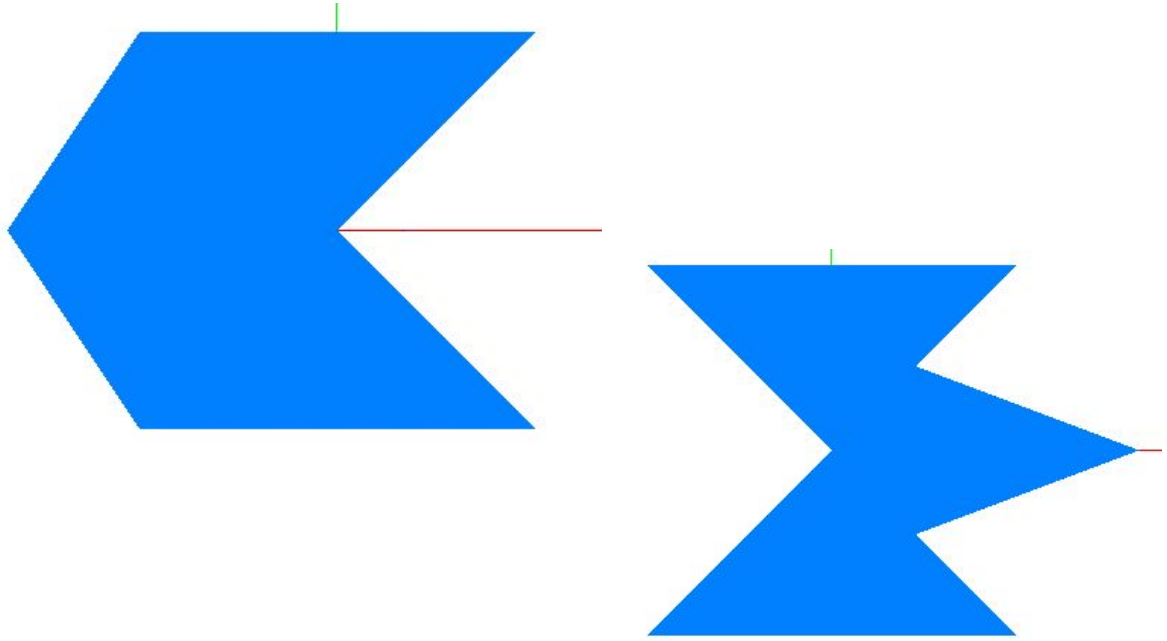
Приклад із полігоном (правильний шостикутник)

```
glBegin(GL_POLYGON);
        glColor3f(0.0,0.5,1.0);
        for(int i=0;i<5;i++)glVertex2f(2.5*cos(0.4*pi*i),2.5*sin(0.4*pi*i));
glEnd();
```



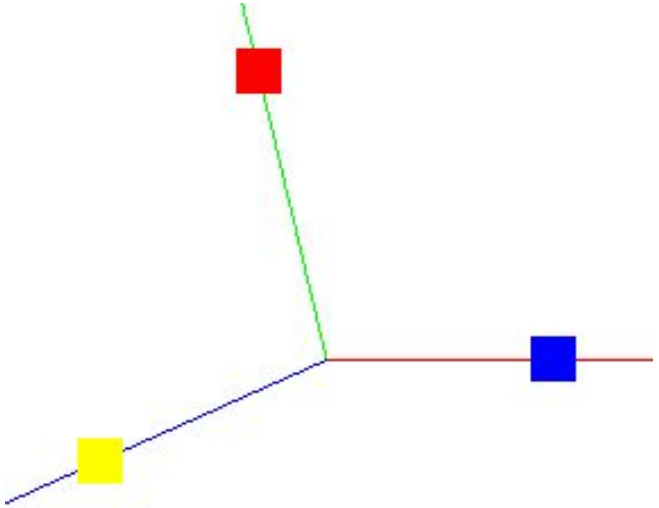
Приклад із полігоном (увігнутий):

```
glBegin(GL_POLYGON);
glColor3f(0.0,0.5,1.0);                    //
glVertex2i(0,0); glVertex2i(3,-3);
glVertex2i(-3,-3); glVertex2i(-5,0);    // glVertex(5,0);
glVertex2i(-3,3); glVertex2i(3,3);
glEnd();
```



Команди відображення графічних примітивів OpenGL

```
// приклад із незалежними точками
glPointSize(20.0);
glBegin(GL_POINTS);
glColor3f(1.0,0.0,0.0); glVertex3i(0,3,0);
glColor3f(1.0,1.0,0.0); glVertex3i(0,0,3);
glColor3f(0.0,0.0,1.0); glVertex3i(3,0,0);
glEnd();
```



```
// приклад із полярною системою координат
glColor3f(1.0,1.0,0.0);        //  жовтий колір
glLineWidth(5.0);                    //  ширина лінії – 5.0
glBegin(GL_LINE_LOOP);    //  відрізки кола
for(int i=0;i<100;i++)
{double angle=2*pi*i/100;
 glVertex2f(5.0*cos(angle),5.0*sin(angle));
}
glEnd();
```