Лабораторна робота № 5

Криві та поверхні у OpenGL. Побудова тривимірних фігур за масивами вершин.

Мета роботи: Вивчити та набути практичних навичок у використанні засобів побудови засобів OpenGL для зображення кривих і поверхонь. Навчитися використовувати функції побудови зображень з використанням масивів атрибутів вершин.

Завдання до роботи: Розробити програму в якій будуть використані засоби OpenGL для зображення кривих і поверхонь. Створити зображення з використанням функцій визначення вершин за масивами атрибутів.

Теоретичні відомості

Криві Безье

Крива Безье задається векторною функцією однієї змінної: C(u) = [X(u), Y(u), Z(u)], де **u** змінюється в деякій області, наприклад, [0, 1].

Фрагмент поверхні Безье задається векторною функцією двох змінних S(u, v) = [X (u, v), Y (u, v), Z (u, v)]. Для кожного значення u і v формула C(u) або S(u, v) обчислює точку на кривій (поверхні). При використанні Безьє-обчислення спочатку вибирають функцію C(u) або S(u, v), включають її (Безье-обчислювач), а потім використовують команду glEvalCoord1 або glEvalCoord2 замість команди glVertex*. У цьому випадку вершина кривої або поверхні може використовуватися точно так само, як і будь-яка інша вершина, наприклад, для формування точки або лінії. Крім того, інші команди автоматично генерують серії вершин, що утворюють простір регулярної однорідної сітки по осі u (або по осях u і v).

Якщо P_i представляє набір контрольних точок, то рівняння виду C(u) = $\sum_{i=0}^{u} B_i^n(u) \times P_i$ - є кривою Безье при зміні u від 0 до 1, де $B_i^n(u)$ це багаточлен Бернштейна ступеня *n*, який задається наступним рівнянням:

$$B_i^n(u) = \binom{n}{i} \times u^i \times (1-u)^{n-i}$$

У складі OpenGL є засоби підтримки роботи з кривими і поверхнями Без'є (так званий Безьє-обчислювач), які дозволяють обчислювати значення поліномів Безьє будь-якого порядку. Безьє обчислювач можна

використовувати для роботи з поліномами від однієї, двох, трьох і чотирьох змінних. Функція обробки полінома однієї змінної налаштовується в процесі ініціалізації OpenGL - програми за допомогою виклику функції:

```
glMap1f(type, u_min, u_max, stride, order, points)
```

Аргумент ***type*** задає тип об'єкта, який буде представлений поліномом Безьє. Можна призначити в якості значення цього аргументу константи, що задають трьох-і чотиривимірні геометричні точки, колір в форматі RGBA, нормалі, індексовані кольори і координати текстур (в діапазоні від одновимірних до чотиривимірних).

Покажчик на масив опорних точок полінома передається функції через аргумент ***points***. Аргументи ***u_min***, ***u_max*** визначають область існування параметра полінома. Аргумент ***stride*** являє собою кількість значень параметра між сегментами кривої. Значення аргументу ***order*** має дорівнювати кількості опорних точок. Для формування кубічної тривимірної кривої у формі B-сплайна, визначеної на інтервалі (0,1), функції glMap1f слід передати такий набір аргументів:

```
points[]= {…};
glMap1f(GL_MAP_VERTEX_3, 0.0, 1.0, 3, 4, data);
```

Після налаштування функція активізується за допомогою виклику: glEnable(type); Якщо функція розрахунку активізована, то можна отримати від неї значення полінома, викликавши функцію: glEvalCoord1f(u). Таким чином, звернення до glEvalCoord1f може замінити звернення до функцій glVertex, glColor, glNormal. Нехай, наприклад, функція розрахунку налаштована на формування кривої Безьє на інтервалі (0,10) за деякого масиву опорних точок. Набір з 100 точок кривої, рівновіддалених на цьому інтервалі можна отримати за допомогою такого фрагмента програми:

```
glBegin(GL_LINE_STRIP)
for( i = 0; i < 100; i++ )
     glEvalCoord1f( i / 100.0f );
glEnd();
```

Якщо значення параметра u розподілені рівномірно, то для обчислення точок на кривій слід використовувати функції glMapGrid1f та glEvalMesh1, наприклад:

```
glMapGrid1f(100, 0.0f, 10.0f);
glEvalMesh1(GL_LINE, 0, 100);
```

Після виклику glMapGrid1f встановлюється рівномірна сітка в 100 відліків, а після виклику функції glEvalMesh1 буде сформована крива.

Наведемо приклад ініціалізації та малювання кривої Безьє:

```
// init
GLfloat points[4][3] = {
     { -4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},
     {2.0, -4.0, 0.0}, {4.0, 4.0, 0.0}};
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &points[0][0]);
glEnable(GL_MAP1_VERTEX_3);
// draw
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_LINE_STRIP);
for (i = 0; i <= 30; i++)
     glEvalCoord1f(i / 30.0f);
glEnd();
glPointSize(5.0);
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_POINTS);
for( i = 0; i < 4; i++ )
     glVertex3fv(&ctrlpoints[i][0]);
glEnd();
```

Поверхні Безьє

Математично фрагмент поверхні Безье визначається рівнянням:

$$S(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)B_j^m(v)P_{ij},$$

де P_{ij} є множиною контрольних точок, а функції B_i многочлени Бернштейна, як і у випадку кривої Безье.

Значення поверхні Безье можуть представляти вершини, нормалі, кольори або текстурні координати. Поверхні Безьє формуються в OpenGL приблизно за тією ж методикою, що й криві, тільки роль функції ініціалізації грає не glMap1*, а glMap2*, відповідно для зчитування результатів слід звертатися до функції glEvalCoord2* замість glEvalCoord1*. В обох функціях потрібно специфікувати дані, які стосуються двох незалежних параметрів ***u*** і ***v***. Наприклад, функція glMap2f має такий формат виклику:

```
glMap2f(type, u_min, u_max, u_stride, u_order,
     v_min, v_max, v_stride, v_order, points);
```

Визначення параметрів для обчислення бікубічної поверхні Безьє, визначеної на області (0,1) x (0,1), виконується наступним чином:

```
glMap2f(GL_MAP_VERTEX_3, 0., 1., 3, 4, 0., 1., 12, 4, points);
```

Для обох незалежних змінних потрібно задати порядок полінома (аргументи ***u_order*** та ***v_order***) і кількість значень параметра між сегментами (аргументи ***u_stride*** та ***v_stride***), що забезпечує додаткову гнучкість при формуванні поверхні. Зверніть увагу на те, що значення ***v_stride*** для другого параметра дорівнює 12, оскільки в масиві опорних точок ***points*** дані зберігаються по рядках. Тому, щоб перейти до наступного елементу цього ж рядка потрібно «переступити» три числа у форматі float, а для переходу до наступного елементу в цьому ж стовпці потрібно «переступити» через 3 * 4 = 12 чисел в форматі float. Спосіб виклику програми розрахунку залежить від того, який результат ми хочемо отримати, - вивести на екран стіку з ліній або сформувати багатокутники для подальшого зафарбовування. Якщо ставиться завдання сформувати на екрані сітку, то відповідний фрагмент програми повинен виглядати приблизно так:

```
for( j = 0; j < 100; j++ )
{
     glBegin( GL_LINE_STRIP );
     for( i = 0; i < 100; i++ )
          glEvalCoord2f( i / 100.0f, j / 100.0f );
     glEnd();
     glBegin( GL_LINE_STRIP );
     for( i = 0; i < 100; i++ )
          glEvalCoord2f( j / 100.0f, i / 100.0f );
     glEnd();
}
```

Якщо потрібно сформувати множину багатокутників, то фрагмент повинен виглядати так:

```
for( j = 0; j < 99; j++ )
{
     glBegin( GL_QUAD_STRIP )
     for( i = 0; i < 100; i++ )
     {
          glEvalCoord2f( i / 100.0f, j / 100.0f);
```

```
            glEvalCoord2f( (i+1) / 100.0f, j / 100.0f);
        }
        glEnd();
    }
```

Для роботи з рівномірною сіткою слід використовувати функції glMapGrid2* та glEvalMesh2. В частині ініціалізації програми потрібно включити наступний фрагмент коду:

```
glMapGrid2f(100, 0.0, 1.0, 100, 0.0, 1.0);
```

У функції малювання потрібно викликати glEvalMesh2:

```
glEvalMesh2(GL_FILL, 0, 100, 0, 100);
```

Для роботи алгоритмів зафарбовування сформованої поверхні при налаштуванні режиму з врахуванням освітлення потрібно додатково викликати функцію glEnable(), передавши їй як аргумент константу GL_AUTO_NORMAL:

```
glEnable (GL_AUTO_NORMAL);
```

Це дозволить OpenGL автоматично обчислювати вектор нормалі до кожної ділянки сформованої поверхні і використовувати цей вектор при зафарбовуванні ділянок цієї поверхні. Приклад елементів програми апроксимації за допомогою поверхні приведено нижче:

```
// init
GLfloat ctrlpoints[6][6][3];
float x, y;
int i, j;
for( i = 0; i < 6; i++ )
{
    x = 3.1415f / 5.0f * i;
    for( j = 0; j < 6; j++ )
    {
        y = 3.1415f / 5.0f * j;
        ctrlpoints[i][j][0]=x;
        ctrlpoints[i][j][1]=y;
        ctrlpoints[i][j][2]=sin(x+y);
    }
}
glMap2f(GL_MAP2_VERTEX_3, 0, 4, 3, 4,
    0, 4, 18, 4, &ctrlpoints[0][0][0]);
glEnable(GL_MAP2_VERTEX_3);
```

```
glEnable(GL_AUTO_NORMAL);

// draw
glEvalMesh2(GL_FILL, 0, 10, 0, 10);
```

Порядок виконання лабораторної роботи

1. Створити програму малювання довільної кривої Безье.
2. Відобразити криву Безье на основі даних заданих в таблиці 1.
3. Додати програмний код для малювання поверхні Безьє з тексту лабораторної роботи.
4. Відобразити довільну поверхню Безьє.
5. Створити тривимірну фігуру з використанням функцій glDrawElements, glNormalPointer, glTexCoordPointer, glVertexPointer.

Таблиця 1.

№	Фігура	№	Фігура
1	Символ нескінченності ∞	6	Символ γ
2	Спіраль	7	Символ §
3	Символ α	8	Символ ղ
4		9	
5		10	

Контрольні питання

1. Математичні основи використання B-сплайнів в комп'ютерній графіці.
2. Інтерполяція і апроксимація з використанням сплайнів.
3. Криві та поверхні Безьє. Математичні основи.
4. Побудова кривих і поверхонь Безьє засобами OpenGL.