POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 3

TEMAT ĆWICZENIA

OpenGL – modelowanie obiektów 3-D

Wykonał:	**Karol Pastewski 252798**
Termin:	**WT/TP 7.30-10.30**
Data wykonania ćwiczenia:	**26.10.2021r.**
Data oddania sprawozdania:	**02.11.2021r.**
Ocena:	

Uwagi prowadzącego:

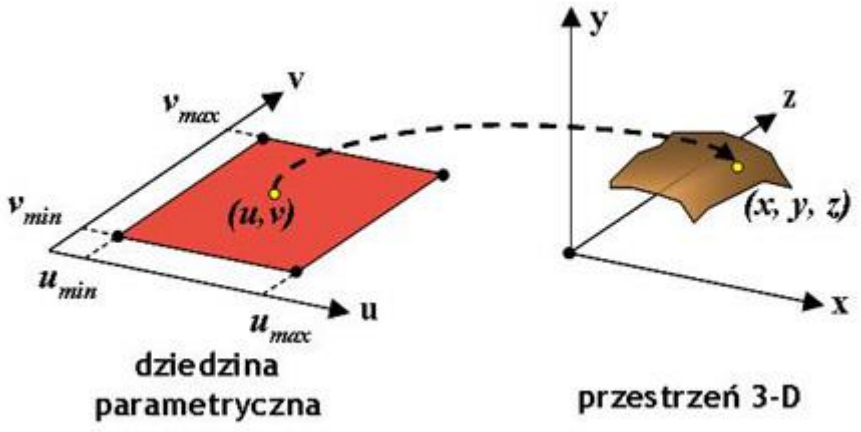
1.Wstęp teoretyczny

1.1. Budowa własnego obiektu 3-D

Modelowanym obiektem będzie jajko, które zostało określone jako powierzchnia opisana równaniami parametrycznymi. Powierzchnię jajka można uzyskać przez odpowiednie obrócenie krzywej Béziera. Wzory opisujące powierzchnię jajka:

$$x(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\cos(\pi v)$$
$$y(u,v) = 160u^4 - 320u^3 + 160u^2 \qquad 0 \le u \le 1$$
$$z(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\sin(\pi v) \qquad 0 \le v \le 1$$

Rysunek 1 Wzory opisujące powierzchnię jajka



Rysunek 2 Przekształcenie dziedziny parametrycznej w powierzchnię w powierzchnię w przestrzeni 3D

Schematyczny opis programu wyświetlającego obiekt 3-D można zapisać tak:

1. Zadać liczbę N, która określać będzie na ile przedziałów podzielony zostanie bok kwadratu jednostkowego dziedziny parametrycznej.
2. Zadeklarować tablicę o rozmiarze NxN, która będzie służyła do zapisywania współrzędnych punktów w przestrzeni 3-D. Każdy element zawiera współrzędne x, y i z jednego punktu.
3. Nałożyć na kwadrat jednostkowy dziedziny parametrycznej równomierną siatkę NxN punktów.
4. Dla każdego punktu u, v nałożonej w kroku poprzednim siatki, obliczyć, przy pomocy podanych wyżej równań, współrzędne x(u, v), y(u, v) i z(u, v) i zapisać je w zadeklarowanej w kroku 2 tablicy.
5. Wyświetlić na ekranie elementy tablicy współrzędnych punktów.

2. Nowe polecenia OpenGL

- `glutKeyboardFunc(void (*func)(unsigned char key, int x, int y))` – polecenie ustawiające funkcję zwrotną obsługującą działanie klawiatury w obecnym oknie. Za każdym razem, gdy użytkownik pisze w oknie programu, generowany jest kod ASCII, który generuje funkcję zwrotną. Zmienne `x` i `y` odpowiadają lokalizacji myszki w momencie, którym klawisz został wciśnięty.
- `glutIdleFunc(void (*func)(void))` – polecenie ustawia funkcje zwrotną dla stanu bezczynności tak, aby program mógł wykonywać zadania w tle albo wyświetlać animację.
- `glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)` – polecenie tworzy rotację o kąt `angle` przez wektor (`x,y,z`).
- `glutWireTeapot(GLDouble size)` – renderuje siatkę obiektu czajnika. Polecenie przyjmuje zmienną `size`, która określa względny rozmiar czajnika
- `glTranslated(GLdouble x, GLdouble y, GLdouble z)` – tworzy translację o punkt (`x,y,z`).

3. Rozwiązanie zadania

```
303   // Funkcja wyświetlająca w konsoli informacje o opcjach programu
304   void initProgram() {
305       std::cout << "Program z laboratorium 3 - modelowanie obiektow 3-D\n";
306       setArray();
307       std::cout << "Odpowiednie klawisze zmieniaja widok modelu:\n";
308       std::cout << "    Wyswietlany obiekt:\n";
309       std::cout << "        'f' - jajko (domyslne)\n";
310       std::cout << "        'g' - czajnik\n";
311       std::cout << "    Model wyswietlania jajka (nie dziala dla obiektu czajnika):\n";
312       std::cout << "        '1' - model chmury punktow (domyslne)\n";
313       std::cout << "        '2' - model siatki\n";
314       std::cout << "        '3' - model wypelnionych trojkatow\n";
315       std::cout << "    Tryb rotacji obiektu:\n";
316       std::cout << "        'q' - brak rotacji (domyslne)\n";
317       std::cout << "        'w' - rotacja po x\n";
318       std::cout << "        'e' - rotacja po y\n";
319       std::cout << "        'r' - rotacja po z\n";
320       std::cout << "        't' - rotacja po x, y i z\n";
321       std::cout << "    Wyswietlanie osi wspolrzednych:\n";
322       std::cout << "        'a' - nie\n";
323       std::cout << "        's' - tak (domyslne)\n";
324       std::cout << "    Operowanie programem:\n";
325       std::cout << "        'esc' - wyjscie z programu\n";
326       std::cout << "Pamietaj, aby aktywnym oknem bylo okno OpenGL'a!!!";
327   }
```

Funkcja `initProgram()` jest wywoływana na początku programu. Wypisuje ona na ekranie konsoli informacje o dostępnych opcjach programu.

```
288   // Funkcja inicjalizuje tablicę 'array' oraz wywołuje funkcje inicjalizujące
289   // kolory i współrzędne punktów
290   void setArray() {
291       std::cout << "Podaj liczbe N = ";
292       std::cin >> N;
293       VERTEXES** newArray = new VERTEXES * [N];
294       for (int i = 0; i < N; i++) {
295           newArray[i] = new VERTEXES[N];
296       }
297       array = newArray;
298       setColors();
299       setVertices();
300   }
```

Funkcja `setArray()` jest wywoływana w `initProgram()`. Inicjalizuje ona tabelę, w której są przechowywane współrzędne punktów oraz ich kolory. Po utworzeniu tabeli wywoływane są funkcje obliczające współrzędne i kolory punktów.

```
247  // Inicjalizacja losowych kolorów dla każdego punktu w tabeli 'array'
248  void setColors() {
249      srand(time(nullptr));       // inicjalizacja generatora losowych liczb
250      float red, green, blue;
251      for (int i = 0; i < N; i++) {
252          for (int j = 0; j < N; j++) {
253              red = (GLfloat)(rand() / (GLfloat)RAND_MAX);        // losowanie liczb
254              green = (GLfloat)(rand() / (GLfloat)RAND_MAX);      // zakres liczb: <0.0; 1.0>
255              blue = (GLfloat)(rand() / (GLfloat)RAND_MAX);       //
256              array[i][j].color3[0] = red;
257              array[i][j].color3[1] = green;
258              array[i][j].color3[2] = blue;
259          }
260      }
261  }
```

Funkcja losuje kolory przez funkcję `rand()` udostępnioną w bibliotece `<time.h>`. Inicjalizacja ziarna generatora liczb losowych(polecenie `srand()`) jest zależne od obecnego czasu systemowego, dzięki czemu program losuje inne liczby dla kolejnych uruchomień programu.

```
263  // Inicjalizacja współrzędnych punktów potrzebnych do wyświetlenia jajka
264  void setVertices() {
265      for (int i = 0; i < N; i++) {
266          float u = (float)i / (N - 1);
267          float uPow5 = pow(u, 5);
268          float uPow4 = pow(u, 4);
269          float uPow3 = pow(u, 3);
270          float uPow2 = pow(u, 2);
271          GLfloat x, y, z;
272
273          y = 160 * uPow4 - 320 * uPow3 + 160 * uPow2;
274
275          for (int j = 0; j < N; j++) {
276              float v = (float)j / (N - 1);
277
278              x = (-90 * uPow5 + 225 * uPow4 - 270 * uPow3 + 180 * uPow2 - 45 * u) * cos(M_PI * v);
279              z = (-90 * uPow5 + 225 * uPow4 - 270 * uPow3 + 180 * uPow2 - 45 * u) * sin(M_PI * v);
280
281              array[i][j].point3[0] = x;
282              array[i][j].point3[1] = y;
283              array[i][j].point3[2] = z;
284          }
285      }
286  }
```

Współrzędne są obliczane ze wzorów podanych na *Rysunku 1*. Wszystkie podane wyżej funkcje są wywoływane tylko raz podczas uruchamiania się programu, przed wyświetleniem się ekranu OpenGL'a. Pomimo tego, że w programie są przeprowadzane rotacje, to współrzędne punktów pozostają stałe w obrębie działania programu (poza początkową inicjalizacją).

```
199  // Funkcja obsługująca rysowanie jajka
200  void Egg() {
201
202      glColor3f(1.0, 1.0, 1.0);    // ustawienie koloru na biały
203      glTranslated(0.0, -(array[(N - 1) / 2][0].point3[1] / 2), 0.0); // wyśrodkowanie obiektu
204
205      if (model == 1) {
206          drawPoints();
207
208      } else if (model == 2) {
209          drawLines();
210      } else {
211          drawTriangles();
212      }
213  }
```

Funkcja `Egg()` ustawia kolor rysowania jajka oraz przesuwa obiekt tak, aby wyglądał jakby był wyśrodkowany na ekranie. Wyśrodkowanie jest dokonane przez przesunięcie obiektu w dół o połowę wysokości najwyższego punktu. Przez specyfikę programu największy y można znaleźć w elemencie o środkowym wierszu (kolumna dowolna), stąd wzięło się wyrażenie widoczne w linii 203. Funkcja `Egg()` wywołuje także funkcje odpowiednie wybranemu modelowi rysowania jajka.

```
126  // Funkcja rysująca każdy punkt w tabeli 'array'
127  void drawPoints() {
128      glBegin(GL_POINTS);
129
130      for (int i = 0; i < N; i++) {
131          for (int j = 0; j < N; j++) {
132              glVertex3fv(array[i][j].point3);
133          }
134      }
135
136      glEnd();
137  }
138
```

Funkcja rysująca punkty na powierzchni jajka.

```
139   // Funkcja rysująca linie pomiędzy punktami w tabeli 'array'
140   void drawLines() {
141
142       for (int i = 0; i < N - 1; i++) {
143           for (int j = 0; j < N - 1; j++) {
144               // linia pionowa
145               glBegin(GL_LINES);
146               glVertex3fv(array[i][j].point3);
147               glVertex3fv(array[i + 1][j].point3);
148               glEnd();
149               // linia pozioma
150               glBegin(GL_LINES);
151               glVertex3fv(array[i][j].point3);
152               glVertex3fv(array[i][j + 1].point3);
153               glEnd();
154
155               // linie ukośne
156               glBegin(GL_LINES);
157               glVertex3fv(array[i][j].point3);
158               glVertex3fv(array[i + 1][j + 1].point3);
159               glEnd();
160               glBegin(GL_LINES);
161               glVertex3fv(array[i][j + 1].point3);
162               glVertex3fv(array[i + 1][j].point3);
163               glEnd();
164           }
165       }
166   }
```

Funkcja rysuje linie łączące punkty na powierzchni jajka. Należy pamiętać, aby nie wyjść poza zakres tablicy dlatego pętle kończą się na przedostatnim elemencie tabeli, a nie na ostatnim (stąd `N - 1`).

```
168   // Funkcja rysująca trójkąty połączone z punktów z tabeli 'array'
169   //
170   // GL_TRIANGLE_FAN - pierwszy wierzchołek jest stały, jest on później
171   // łączony z grupami dwóch kolejnych wierzchołków w trójkąt, czyli np.
172   // dla 5 wierzchołków wyjdą trójkąty (0, 1, 2) i (0, 3, 4)
173   void drawTriangles() {
174       for (int i = 0; i < N - 1; i++) {
175           for (int j = 0; j < N - 1; j++) {
176               glBegin(GL_TRIANGLE_FAN);
177
178               glColor3fv(array[i][j].color3);
179               glVertex3fv(array[i][j].point3);
180
181               glColor3fv(array[i][j + 1].color3);
182               glVertex3fv(array[i][j + 1].point3);
183
184               glColor3fv(array[i + 1][j + 1].color3);
185               glVertex3fv(array[i + 1][j + 1].point3);
186
187               glColor3fv(array[i + 1][j].color3);
188               glVertex3fv(array[i + 1][j].point3);
189
190               glColor3fv(array[i + 1][j + 1].color3);
191               glVertex3fv(array[i + 1][j + 1].point3);
192
193               glEnd();
194           }
195       }
196
197   }
```

Funkcja rysująca trójkąty. Należy pamiętać o zakresie tablicy tak jak w funkcji wyżej opisanej. To tworzenia trójkątów pomocny jest prymityw z biblioteki OpenGL `GL_TRIANGLE_FAN`, który tworzy trójkąt z połączenia pierwszego wierzchołka oraz parą dwóch kolejnych wierzchołków.

```
52  // Funkcja obliczająca obecny stan obrotu
53  void calcRotation() {
54      switch (rotationMode) {
55      case 1:
56          currentRotation[0] += 0.5;
57          if (currentRotation[0] > 360.0) currentRotation[0] -= 360.0;
58          break;
59
60      case 2:
61          currentRotation[1] += 0.5;
62          if (currentRotation[1] > 360.0) currentRotation[1] -= 360.0;
63          break;
64
65      case 3:
66          currentRotation[2] += 0.5;
67          if (currentRotation[2] > 360.0) currentRotation[2] -= 360.0;
68          break;
69
70      case 4:
71          currentRotation[0] += 0.5;
72          if (currentRotation[0] > 360.0) currentRotation[0] -= 360.0;
73          currentRotation[1] += 0.5;
74          if (currentRotation[1] > 360.0) currentRotation[1] -= 360.0;
75          currentRotation[2] += 0.5;
76          if (currentRotation[2] > 360.0) currentRotation[2] -= 360.0;
77          break;
78      default:
79          break;
80      }
81
82      glutPostRedisplay();        // odświeżenie zawartości aktualnego okna
83  }
```

Program ma możliwość zastosowania rotacji obiektu, dlatego do obliczania obecnego stanu obrotu jest używana powyższa funkcja. Zależnie od opcji wybranej w programie funkcja zmienia obecny stan obrotu obiektu.

```
329  // Funkcja obsługująca działanie programu za pomocą klawiszy klawiatury
330  void keys(unsigned char key, int x, int y) {
331      if (objectMode == 1) {
332          if (key == '1') model = 1;
333          if (key == '2') model = 2;
334          if (key == '3') model = 3;
335      }
336      if (key == 'q') rotationMode = 0;
337      if (key == 'w') rotationMode = 1;
338      if (key == 'e') rotationMode = 2;
339      if (key == 'r') rotationMode = 3;
340      if (key == 't') rotationMode = 4;
341      if (key == 'a') showAxes = 0;
342      if (key == 's') showAxes = 1;
343      if (key == 'f') objectMode = 1;
344      if (key == 'g') objectMode = 2;
345      if (key == (char)27) exit(0);
346
347      renderScene(); // przerysowanie obrazu sceny
348  }
```

Funkcja obsługująca działanie klawiatury.