POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Grafika komputerowa i komunikacja człowiek - komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 4

TEMAT ĆWICZENIA

OpenGL – interakcja z użytkownikiem

Wykonał:	**Karol Pastewski 252798**
Termin:	**WT/TP 7.30-10.30**
Data wykonania ćwiczenia:	**09.11.2021r.**
Data oddania sprawozdania:	**16.11.2021r.**
Ocena:	

Uwagi prowadzącego:

1. Wstęp teoretyczny

1.1. Rzut perspektywiczny

W poprzednich ćwiczeniach do wyświetlenia obiektu 3-D używaliśmy rzutu ortograficznego. W tym rzucie rzutnia, czyli płaszczyzna na której powstawał obraz, była równoległa do płaszczyzny tworzonej przez osie x i y, a proste rzutowania biegły równolegle do osi z. Aby umożliwić pokazanie efektów przemieszczeń obiektu we wszystkich osiach należy zastosować rzutowanie perspektywiczne. Rzut perspektywiczny jest lepszy od równoległego nie tylko ze względu na możliwość prezentacji przemieszczeń, pozwala także lepiej pokazać na płaszczyźnie geometrii trójwymiarowego obiektu.

2. Nowe polecenia OpenGL

- **glutMouseFunc**(void (*func)(int buton,
  int state,
  int x,
  int y));

  Polecenie ustawiające funkcję zwrotną obsługującą działanie przycisków myszy. Parametr `button` odpowiada za rodzaj wciśniętego przycisku (`GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON` i `GLUT_RIGHT_BUTTON`). Parametr `state` może przyjąć wartość `GLUT_DOWN`, kiedy przycisk zostaje naciśnięty albo `GLUT_UP`, kiedy przycisk zostaje zwolniony. Parametry `x` i `y` wskazują na względne współrzędne okna po zmianie stanu przycisku myszy.
- **glutMotionFunc**(void (*func)(int x,
  int y));

  Polecenie ustawiające funkcję zwrotną obsługującą ruch myszki. Funkcja zwrotna jest wywoływana za każdym razem, gdy jeden albo więcej przycisków na myszy zostanie wciśnięty. Parametry `x` i `y` wskazując na lokacje kursora myszy na oknie programu.
- **gluPerspective**(GLdouble fovy,
  GLdouble aspect,
  GLdouble zNear,
  GLdouble zFar);

  Polecenie służy do definiowania obszaru (bryły) widoczności dla rzutu perspektywicznego. Parametr `fovy` określa kąt widzenia, w stopniach, dla współrzędnej `y`. Parametr `aspect` określa kąt widzenia, w stopniach, dla współrzędnej `x`. Jest to proporcja pomiędzy szerokością (`x`) i wysokością (`y`). Parametry `zNear` oraz `zFar` określają zakres dla kierunku `z`, dla którego obiekt będzie wyświetlony. Jeżeli obiekt będzie miał współrzędną `z` wychodzącą poza zakres określony przez `zNear` i `zFar`, to nie zostanie on wyświetlony.
- **gluLookAt**(GLdouble eyeX,
  GLdouble eyeY,
  GLdouble eyeZ
  GLdouble centerX,
  GLdouble centerY,
  GLdouble centerZ,
  GLdouble upX,
  GLdouble upY
  GLdouble upZ);

  Polecenie pozwalające na definiowanie położenia obserwatora względem punktu. Parametry `eyeX`, `eyeY` oraz `eyeZ` określają pozycję, w której znajduje się obserwator. Parametry `centerX`, `centerY` oraz `centerZ` określają punkt, na który obserwator ma patrzeć. Parametry `upX`, `upY` oraz `upZ` określa kierunek wektora up. Aby obserwator był zwrócony w stronę punktu `upY` musi mieć dodatnią wartość.

3. Rozwiązanie zadania

```
413  // Funkcja obsługuje wciśnięcie przycisku na myszcze
414  void Mouse(int btn, int state, int x, int y) {
415      if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
416          x_pos_old = x;         // przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
417          y_pos_old = y;
418          status = 1;          // wciśnięty został lewy klawisz myszy
419      } else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
420          y_pos_old = y;
421          status = 2;
422      } else {
423          status = 0;          // nie został wciśnięty żaden klawisz
424      }
425  }
426
427  // Funkcja "monitoruje" położenie kursora myszy i ustawia wartości odpowiednich
428  // zmiennych globalnych
429  void Motion(GLsizei x, GLsizei y) {
430
431      delta_x = x - x_pos_old;     // obliczenie różnicy położenia kursora myszy
432      delta_y = y - y_pos_old;
433
434      x_pos_old = x;             // podstawienie bieżącego położenia jako poprzednie
435      y_pos_old = y;
436
437      glutPostRedisplay();     // przerysowanie obrazu sceny
438  }
```

Funkcje obsługujące działanie przycisków oraz ruchu myszki.

```
234  // Funkcja obsługująca rysowanie czajnika
235  void Teapot() {
236      if (status == 1) {       // jeśli lewy klawisz myszy wciśnięty
237          teapotRotation[0] +=delta_x * pix2angle;
238          teapotRotation[1] +=delta_y * pix2angle;
239
240          if (teapotRotation[0] > 360.0) teapotRotation[0] -= 360.0;
241          else if(teapotRotation[0] < -360.0) teapotRotation[0] += 360.0;
242          if (teapotRotation[1] > 360) teapotRotation[1] -= 360.0;
243          else if(teapotRotation[1] < -360.0) teapotRotation[1] += 360.0;
244
245      } else if (status == 2) {        // jeśli prawy klawisz myszy wciśnięty
246          viewerTeapotZ += delta_y / 10;
247          if (viewerTeapotZ > 20) {
248              viewerTeapotZ = 20;
249          } else if (viewerTeapotZ < 2) {
250              viewerTeapotZ = 2;
251          }
252      }
253
254      gluLookAt(0.0, 0.0, viewerTeapotZ, centerX, centerY, centerZ, 0.0, 1.0, 0.0);
255      // Zdefiniowanie położenia obserwatora
256
257      glRotatef(teapotRotation[0], 0.0, 1.0, 0.0);
258      glRotatef(teapotRotation[1], 1.0, 0.0, 0.0);
259
260      if (showAxes == 1) drawAxes();
261
262      glColor3f(1.0, 1.0, 1.0);     // ustawienie koloru na biały
263      glutWireTeapot(4.0);
264  }
```

Funkcja rysuje czajnik na ekranie w rzucie perspektywicznym obsługujący ruch wokół obiektu za pomocą obracania czajnika w miejscu. Punkt, na który jest zwrócony obserwator się nie zmienia i tak samo położenie obserwatora. Linie od 240 do 243 zapewniają, że wartości kątów obrotu zawsze są pomiędzy 0, a 360. Użytkownik ma zablokowaną możliwość przybliżania się do obiektu do pewnego momentu, aby zapobiec niepożądanemu działaniu programu.

```
196  // Funkcja obsługująca rysowanie jajka
197  void Egg() {
198
199      if (status == 1) {                    // jeśli lewy klawisz myszy wciśnięty
200          theta += (delta_x * pix2angle) / 100;     // modyfikacja kąta obrotu o kąt proporcjonalny
201          phi += (delta_y * pix2angle) / 100;
202
203          if (theta > 360.0) theta -= 360.0;
204          else if(theta < -360.0) theta += 360.0;
205          if (phi > 360) phi -= 360.0;
206          else if(phi < -360.0) phi += 360.0;
207
208      } else if (status == 2) {         // jeśli prawy klawisz myszy wciśnięty
209          radius += delta_y / 10;
210          if (radius > 20) {
211              radius = 20;
212          } else if (radius < 2) {
213              radius = 2;
214          }
215      }
216      viewer[0] = radius * cos(theta) * cos(phi);
217      viewer[1] = radius * sin(phi);
218      viewer[2] = radius * sin(theta) * cos(phi);
219
220      gluLookAt(viewer[0], viewer[1], viewer[2], centerX, centerY, centerZ, 0.0, 1.0, 0.0);
221      // Zdefiniowanie położenia obserwatora
222
223      if (showAxes == 1) drawAxes();
224
225      glColor3f(1.0, 1.0, 1.0);     // ustawienie koloru na biały
226      glTranslated(0.0, -(ARRAY[(N - 1) / 2][0].point3[1] / 2), 0.0);     // wyśrodkowanie obiektu
227
228      if (model == 1) drawPoints();
229      else if (model == 2) drawLines();
230      else drawTriangles();
231
232  }
```

Funkcja rysuje czajnik na ekranie w rzucie perspektywicznym obsługujący ruch wokół obiektu za pomocą obracania obserwatora wokół obiektu. Punkt, na który jest zwrócony obserwator się nie zmieni, ale położenie obserwatora już tak. Linie od 203 do 206 zapewniają, że wartości kątów obrotu zawsze są pomiędzy 0, a 360. Linie od 216 do 218 obliczają nowe położenie obserwatora według wzorów przedstawionych na Rysunku 1. Użytkownik ma zablokowaną możliwość przybliżania się do obiektu do pewnego momentu, aby zapobiec niepożądanemu działaniu programu.

$$x_s(\Theta,\Phi) = R\cos(\Theta)\cos(\Phi)$$

$$y_s(\Theta,\Phi) = R\sin(\Phi)$$

$$z_s(\Theta,\Phi) = R\sin(\Theta)\cos(\Phi)$$

$$0 \le \Theta \le 2\pi$$

$$0 \le \Phi \le 2\pi$$

Rysunek 1 Wzory określające położenie obserwatora