Grafika komputerowa i komunikacja człowiek-komputer - labolatorium 3

Kacper Małkowski 252724

prowadzący - dr. inż. Jan Nikodem

1 Wstęp

Ćwiczenie ma za zadanie pokazać, jak przy pomocy funkcji biblioteki OpenGL z rozszerzeniem GLUT można zrealizować prostą interakcję, polegające na sterowaniu ruchem obiektu i położeniem obserwatora w przestrzeni 3D. Do sterowania służyla będzie mysz. Ponadto zostaną zilustrowanie sposoby prezentacji obiektów trójwymiarowych w rzucie perspektywicznym.

2 Plan działania

Zadaniami do zrobienia były dwa podejścia do obrotu i przybliżenia/oddalenia obiektu. Pierwszym podejsciem jest obracanie obiektu, a drugim przemieszczanie punktu widzenia. W zadaniach obiektem testowanym jest czajnik i jajko z poprzedniego ćwiczenia.

3 Program

Poniżej opiszę wszystkie fragmenty kodu, który napisałem lub zmodyfikowałem. Jako szkielet do napisania programu posłużyły mi przykłady ze strony ćwiczenia. Reszta kodu wraz z komentarzami, została przesłana mailem.

3.1 Zmienne globalne

Poniżej wypisane zostały wszystkie zmienne globalne używane w moim programie.

```
//zmienne ogolne
typedef float point3[3];
int testedObject = 1; //rysowany obiekt
int task = 1;  //wybrane zadanie

//zmienne jajka
int n = 40;//ilosc punktow
float scale = 3.0;//wielkosc obiektu
point3** points;//siatka punktow
point3** color; //kolory

//zmienne zadanie 1 i 2
static GLfloat theta[] = { 0.0, 0.0, 0.0 };
static GLfloat pix2angle;     // przelicznik pikseli na stopnie
static GLint statusLeft = 0;       // stan klawisza lewego
static GLint statusRight = 0;// stan klawisza prawego
static int x_pos_old = 0;       // pozycje kursora myszy
static int delta_x = 0;
static int y_pos_old = 0;
static int delta_y = 0;

//zmienne zadanie 2
```

```
23 float rViewer = 10;//R wokol punktu obserwowanego
24 float rObject = 0;//R punktu obserwowanego (uzyte w celu mozliwosci przesuwania
       tegoz punktu)
25 static GLfloat viewer[] = { 0.0, 0.0, 10.0 };//pozycja punktu widzenia
26 static GLfloat object[] = { 0.0, 0.0, 0.0 };//pozycja punktu obserwowanego
27 static GLfloat azymuth = 0;     //katy obserwacji punktu obserwowanego
28 static GLfloat elevation = 0;
```

3.2 main()

Funkcja main w porównaniu do szablonu ze strony zsk zyskała funkcje monitorujące mysz, kursor i klawiaturę (linijki 28-30). Przy jakichkolwiek akcjach na wymienionych mediach tj. stuknięcie w klawisz, przesunięcie myszy, albo przyciśnięcie przycisku na myszy wywoła odpowiednią funkcję. Dodałem także fragment kodu z poprzedniego ćwiczenia. Kod ten służy do generacji siatki na pozniejsze współrzędne punktów na powierzchi jajka (linijki 16-19), oraz losowanie kolorów przypisanych do tych punktów (linijki 3-14). Ostanią zmianą jest dodanie prostego wyświetlania tekstu w celu wytłumaczenia obsługi programu (linijka 21).

```
1  void main(void)
2  {
3    srand(time(NULL));
4    color = new point3 * [n + 1];//tablica kolorow
5    for (int i = 0; i <= n; i++) {
6      color[i] = new point3[n + 1];
7    }
8    for (int i = 0; i <= n; i++) {//losowanie kolorow
9      for (int ii = 0; ii <= n; ii++) {
10       color[i][ii][0] = (rand() % 101) * 0.01;
11       color[i][ii][1] = (rand() % 101) * 0.01;
12       color[i][ii][2] = (rand() % 101) * 0.01;
13     }
14   }
15
16   points = new  point3 * [n + 1];//tablica punktow
17   for (int i = 0; i <= n; i++) {
18     points[i] = new point3[n + 1];
19   }
20
21   cout << "zadanie 1: kliknij 1\nzadanie 2: kliknij 2\ntestuj czajnik: kliknij
         c\ntestuj jajko: kliknij j\njesli podczas zoomowania w 2 zadaniu utknales:
         kliknij r, aby scentrowac na obiekcie" << endl;
22   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
23   glutInitWindowSize(1000, 1000);
24   glutCreateWindow("Rzutowanie perspektywiczne");
25   glutDisplayFunc(RenderScene);
26   glutReshapeFunc(ChangeSize);
27   MyInit();
28   glutMouseFunc(Mouse);//"lapanie" akcji na przyciskach myszy
29   glutMotionFunc(Motion);//"lapanie" ruchu myszki
30   glutKeyboardFunc(keys);//"lapanie" akcji na klawiaturze
31   glEnable(GL_DEPTH_TEST);
32   glutMainLoop();
33 }
```

3.3 keys()

Funkcja ta wywołuje się zawsze, gdy naciśnięty zostanie jakiś klawisz. Obsługiwane przez program klawisze to: 1, 2, c, j, r. Kliknięcie 1 lub 2 zmienia wyswietlane zadanie, oraz resetuje zmienne uzywane w poprzednim zadaniu. C i j określają jaki obiekt zostanie wyswietlony. Natomiast r resetuje zooma w 2 drugim zadaniu (zagadnienie zostanie zgłębione w punkcie 3.8 sprawozdania).

```
1 void keys(unsigned char key, int x, int y)
2 {
```

```
  if (key == 'j') {//wyswietlanie jajka
    obiekt = 2;
  }
  if (key == 'c') {//wyswietlenie czajnika
    obiekt = 1;
  }
  if (key == '1') {//uruchomienie 1 zadania
    if (zadanie == 1) {
      return;
    }
    rViewer = 10;//reset zmiennych z 2 zadania
    viewer[0] = 0.0;
    viewer[1] = 0.0;
    viewer[2] = 10.0;
    object[0] = 0.0;
    object[1] = 0.0;
    object[2] = 0.0;
    zadanie = 1;
  }
  if (key == '2') {//uruchomienie 2 zadania
    if (zadanie == 2) {
      return;
    }
    rViewer = 10;//reset zmiennych z 1 zadania
    azymuth = M_PI / 4;
    elevation = M_PI / 4;
    scale = 3.0;
    zadanie = 2;
  }
  if (key == 'r') {//reset ustawienia zooma w 2 zadaniu
    rViewer = 10;
  }

  RenderScene();
}
```

3.4 mouse()

Funkcja ta wywołuje się zawsze, gdy naciśnięty zostanie klawisz na myszy. Przy naciśnięciu klawisza prawego ustawiana jest flaga statusRight, która informuje program, że przycisk jest wciśnięty. Przy naciśnięciu klawisza lewego ustawiana jest flaga statusLeft, która informuje program, że przycisk jest wciśnięty.

```
void Mouse(int btn, int state, int x, int y)
{
  if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN){//sprawdzenie czy przycisniety
      zostal prawy klawisz
    y_pos_old = y;
    statusRight = 1;          //ustawienie flagi przycisku
  }
  else {
    statusRight = 0;     //ustawienie flagi przycisku
  }

  if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN){//sprawdzenie czy przycisniety
      zostal lewy klawisz
    x_pos_old = x;
    y_pos_old = y;
    statusLeft = 1;     //ustawienie flagi przycisku
  }
  else {
    statusLeft = 0;    //ustawienie flagi przycisku
  }
  RenderScene();
}
```

3.5 motion()

Funkcja ta wywołuje się zawsze, gdy pozycja kursora się zmieni. Ustawia ona zmienne globalne na aktualne pozycje myszy, oraz oblicza zmianę pozycji, która jest uzywana do obliczania kątów obrotów obiektów testowanych. W linijce 9 następuje odświeżenie okna.

```
1  void Motion(GLsizei x, GLsizei y)
2  {
3    delta_x = x - x_pos_old;//zmiana pozycji x
4    x_pos_old = x;
5
6    delta_y = y - y_pos_old;//zmiana pozycji y
7    y_pos_old = y;
8
9    glutPostRedisplay();
10 }
```

3.6 RenderScene()

Funkcja ta jest głównym elementem programu. Wywołuje ona konkretne zadanie do wyświetlenia, ustawia punkt widzenia, oraz wyświetla przygotowny obraz. Zmienne viewer[] i object[] to współrzędne punktu widzenia i punktu obserwowanego. W zadaniu 1 są one stałe (ustawiają się przy zmianie zadania w funkcji keys(), w podpunkcie 3.3 sprawozdania). W zadaniu 2 zmienne te zmieniają się w trakcie działania (więcej w punkcie 3.8 sprawozdania).

```
1  void RenderScene(void)
2  {
3    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);//deklaracja bufferow
4    glLoadIdentity();
5    gluLookAt(viewer[0], viewer[1], viewer[2], object[0], object[1], object[2], 0.0,
         1.0, 0.0);//ustawienie pozycji punktu widzenia i pozycji punktu obserwowanego
6
7    switch (zadanie) {//wywolanie odpowiedniego zadania
8    case 1:
9      zadanie1();
10     break;
11   case 2:
12     zadanie2();
13     break;
14   }
15   glutSwapBuffers();//zmiana bufferow i wyswietlanie
16 }
```

3.7 zadanie1()

Funkcja ta obsługuje zadanie 1, w którym zgodnie z opisem w punkcie 2 sprawozdania działa na zasadzie obrotu i przeskalowania obiektu wyświetlanego. Obrót (linijki 8-9) następuje jako zmiana kąta obiektu (linijki 4-7), który zmienia się, gdy lewy klawisz jest wciśnięty. Zoom działa jako zwiększanie lub zmniejszanie obiektu (linijki 11-20) (dla jajka zmiana scale zmienia pozycje punktów siatki mnożąc wszystkie współrzędne przez scale, więcej na ten temat w kodzie źródłowym).

```
1  void zadanie1() {//obiekt jest obracany a zoom jest wykonany jako
       poweikszanie/pomnijeszanie obiektu
2    Axes();
3
4    if (statusLeft == 1) {//obrot
5      theta[0] += delta_x * pix2angle;
6      theta[1] += delta_y * pix2angle;
7    }
8    glRotatef(theta[0], 0.0, 1.0, 0.0);
9    glRotatef(theta[1], 1.0, 0.0, 0.0);
10
11   if (statusRight == 1) {//zoom
```

```
12      if (delta_y < 0) {
13        scale += 0.1;
14      }
15      else {
16        scale -= 0.1;
17      }
18      if (scale <= 0.1) {
19        scale = 0.1;
20      }
21    }
22
23    switch (obiekt) {//wybranie obiektu
24    case 1:
25      glColor3f(1.0f, 1.0f, 1.0f);
26      glutWireTeapot(scale);
27      break;
28    case 2:
29      jajko();
30      break;
31    }
32  }
```

3.8 zadanie2()

Funkcja ta obsługuje zadanie 1, w którym zgodnie z opisem w punkcie 2 sprawozdania działa na zasadzie zmiany położenia punktu widzenia. Pozycja punktu widzenia obliczana jest ze wzorów ze strony zsk (linijki 24-26). Zmiana pozycji następuje poprzez zmianę kątów azymuth i elevation (linijki 3-6), gdy jest wciśniety lewy przycisk myszy. W wyniku rozmowy pod koniec zajęć zmieniłem sposób zoomowania, tak by możliwe było wykonanie tej operacji poza punkt (0,0,0). W tym celu uruchomiłem przesuwanie punktu obserwowanego, którego współrzędne obliczane są w taki sposób (linijki 18-21), aby punkt ten znajdował sie na prostej przecinającej punkt widzenia i środek obiektu, oraz utrzymywał stałą odległość od punktu widzenia równą 10. Dzięki takiemu podejsciu nawet, gdy podczas zoomowania punkt widzenia znajdzie się w punkcie (0,0,0), punkt obserwowany jest przesunięty i jest możliwość daleszego przybliżania. Czasem przyblizając za bardzo wyjdziemy poza obiekt z drugiej strony, w takim przypadku aby wrócić wystarczy kliknąć r i zostanie zresetowana pozycja punktu obserwowanego na współrzędne (0,0,0).

```
1  void zadanie2() {//przesuwany jest punkt widzenia, a zoom jest wykonany jako
       przesuwanie go w strone punktu obserwowanego
2    Axes();
3    if (statusLeft == 1) {//obrot
4      azymuth += delta_x * pix2angle / 100;
5      elevation += delta_y * pix2angle / 100;
6    }
7
8    if (statusRight == 1) {//zoom
9      if (delta_y > 0) {
10       rViewer += 0.2;
11     }
12     else {
13       rViewer -= 0.2;
14     }
15   }
16
17
18   rObject = rViewer - 10;//obliczenie pozycji punktu obserwowanego
19   object[0] = rObject * cos(azymuth) * cos(elevation);
20   object[1] = rObject * sin(elevation);
21   object[2] = rObject * sin(azymuth) * cos(elevation);
22
23
24   viewer[0] = rViewer * cos(azymuth) * cos(elevation);//obliczenie pozycji punktu
       widzenia
25   viewer[1] = rViewer * sin(elevation);
```

```
26    viewer[2] = rViewer * sin(azymuth) * cos(elevation);
27
28
29    switch (obiekt) {//wybranie obiektu
30    case 1:
31      glColor3f(1.0f, 1.0f, 1.0f);
32      glutWireTeapot(scale);
33      break;
34    case 2:
35      scale = 3.0;
36      jajko();
37      break;
38    }
39  }
```

4 Wnioski

Używanie w programie klawiatury, myszki i jej przycisków, nie należy do najtrudniejszych zadań. Funkcje wydają się względnie proste i intuicyjne. Największym problemem podczas pisania programu, było stworzenie systemu przybliżania/oddalania w zadaniu 2. Rozmowa pod koniec zajęć zburzyła łatwe podejscie i zmusiła do przemyślenia rozwiązania.