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:	**Bartosz Szymczak, nr indeksu 252734**
Termin:	**Poniedziałek TN 7:30-10:30**
Data wykonania ćwiczenia:	**29.11.2021r.**
Data oddania sprawozdania:	
Ocena:	

Uwagi prowadzącego:

Spis treści

1. Cel ćwiczenia

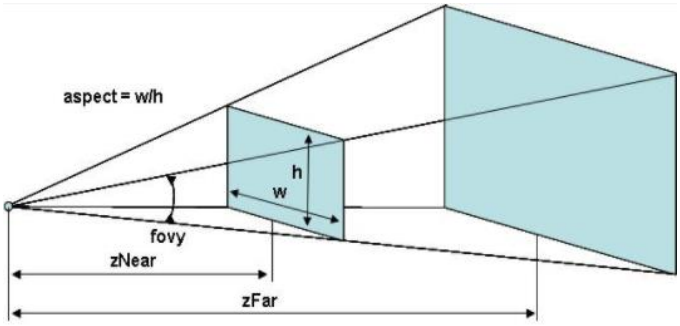
Celem ćwiczenia jest realizacja prostej interakcji, polegającej na sterowaniu ruchem obiektu i położeniem obserwatora w przestrzeni trójwymiarowej. Do sterowania będzie służyła mysz. Dodatkowo zaprezentowane zostaną sposoby prezentacji obiektów trójwymiarowych w rzucie perspektywicznym.

Pierwszym zadaniem jest program wyświetlający czajnik przy użyciu gotowej funkcji. Powinien być rysowany w położeniu początkowym. Przy wciśniętym lewym przycisku myszy oraz ruchu kursora myszy w kierunku poziomym, czajnik powinien obracać się wokół osi y. Przy wciśniętym lewym przycisku myszy oraz ruchu kursora w kierunku pionowym, czajnik powinien obracać się wokół osi x. Przy wciśniętym prawym przycisku myszy oraz ruchu kursora myszy w kierunku pionowym, obserwator powinien się zbliżać i oddalać od czajnika.

Drugim zadaniem jest program wyświetlający jajko stworzone w poprzednim ćwiczeniu. Należy go rozwinąć o funkcje pozwalające na zmianę azymutu kąta Θ przy wciśniętym lewym przycisku myszy i poziomym ruchu kursora oraz zmianę azymutu kąta elewacji ϕ przy wciśniętym lewym przycisku myszy i pionowym ruchu kursora. Prawy przycisk myszy oraz ruch kursora w kierunku pionowym powinien zmieniać promień.

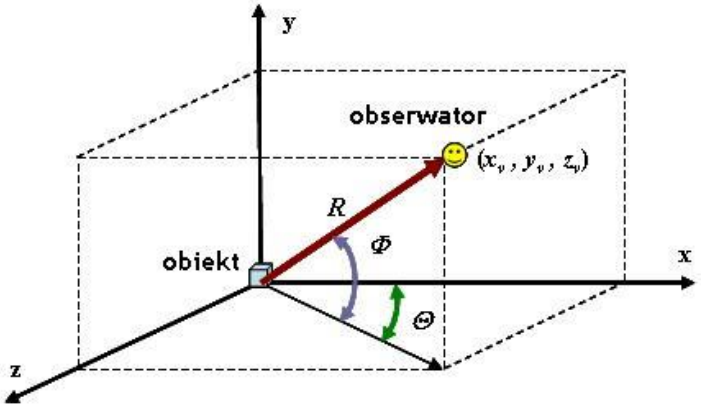
2. Wstęp teoretyczny

W celu umożliwienia pokazania efektów przemieszczeń obiektu we wszystkich osiach należy zastosować rzutowanie perspektywiczne. Efekt obracania obiektu można wykonać dwoma sposobami. Obiekt można obracać wokół osi przy ustalonym położeniu obserwatora, co zostanie wykorzystane w zadaniu pierwszym. Widok bryły i interpretacja poszczególnych argumentów funkcji pokazuje rys. 1. Dwa widoczne na nim prostokąty określają granice w jakich elementy wchodzące w skład opisu sceny będą widoczne na obrazie. Fragmenty sceny leżące przed mniejszym i za większym prostokątem nie zostaną wyświetlone.



Rysunek 1. Bryła widoczności dla rzutu perspektywicznego

Drugim sposobem jest obracanie obserwatorem wokół obiektu, co zostanie wykorzystane w zadaniu drugim. Sterowanie położeniem obserwatora zostanie zrealizowane przy pomocy dwóch kątów. Pierwszy (azymut Θ) określa kierunek patrzenia na obiekt, a drugi (elewacja ϕ) określa pośrednio wysokość położenia obserwatora nad hipotetycznym horyzontem. Idea została ukazana na Rysunku 2.



Rysunek 2. Obiekt, obserwator i kąty azymutu i elewacji

Na podstawie powyższego rysunku można łatwo wyznaczyć zależności wiążące położenie obserwatora z azymutem, kątem elewacji i promieniem sfery powierzchni, na której znajduje się obserwator.

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

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

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

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

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

3. Opis programu

Programy zostały napisane w środowisku Microsoft Visual Studio, użyto bibliotek OpenGL oraz GLUT. Wykorzystano język C++, na bazie szkieletu programu z instrukcji.

3.1 Zadanie pierwsze

3.1.1 Biblioteki i zmienne globalne

```
#define _USE_MATH_DEFINES
#include <windows.h>
#include <iostream>
#include <gl/gl.h>
#include <gl/glut.h>
#include <cmath>
using namespace std;

//Definicja zmiennej przechowującej współrzędne (x,y,z)
typedef float point3[3];

//Inicjalizacja współrzędnych położenia obserwatora
static GLfloat viewer[] = { 0.0f, 0.0f, 10.0f };

//Kąt obrotu obiektu,
static GLfloat theta[3] = { 0.0f, 0.0f, 0.0f };

//Przelicznik pikseli na stopnie
static GLfloat pix2anglex, pix2angley;

//Stan klawiszy myszy
// 0 - nie naciśnięto żadnego klawisza
// 1 - naciśnięty został lewy klawisz myszy
// 2 - naciśnięty został prawy klawisz myszy
static GLint status = 0;

//Poprzednia pozycja kursora myszy
static int x_pos_old = 0;

//Różnica pomiędzy pozycją bieżącą i poprzednią kursora myszy
static int delta_x = 0;

// poprzednia pozycja kursora myszy
static int y_pos_old = 0;

//Różnica pomiędzy pozycją bieżącą i poprzednią kursora myszy
static int delta_y = 0;
```

3.1.2 Funkcja main

```
// Główny punkt wejścia programu.
void main(void)
{
   //Wyświetlenie informacji o programie w oknie konsolowym
   info();
   //Funkcja ustawia tryby wyświetlania:
   // GLUT_DOUBLE - używanie podwójnego buforu przestrzeni renderowania,
   // GLUT_RGB - używanie kolorów w konwencji RGB
   // GLUT_DEPTH - użycie buforu do stworzenia kontroli głębi
```

```
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

   //Ustawienie początkowej wielkości okna aplikacji
   glutInitWindowSize(300, 300);

   //Tworzenie okna aplikacji o podanym tytule
   glutCreateWindow("Rzutowanie perspektywiczne");

   // Ustala funkcję zwrotną odpowiedzialną za badanie stanu myszy
   glutMouseFunc(Mouse);

   // Ustala funkcję zwrotną odpowiedzialną za badanie ruchu myszy
   glutMotionFunc(Motion);

   // Określenie, że funkcja RenderScene będzie funkcją zwrotną
   // (callback function).  Będzie ona wywoływana za każdym razem
   // gdy zajdzie potrzeba przerysowania okna
   glutDisplayFunc(RenderScene);

   // Dla aktualnego okna ustala funkcję zwrotną odpowiedzialną
   // za zmiany rozmiaru okna
   glutReshapeFunc(ChangeSize);

   // Funkcja MyInit() (zdefiniowana powyżej) wykonuje wszelkie
   // inicjalizacje konieczne  przed przystąpieniem do renderowania
   MyInit();

   // Włączenie mechanizmu usuwania niewidocznych elementów sceny
   glEnable(GL_DEPTH_TEST);

   // Funkcja uruchamia szkielet biblioteki GLUT
   glutMainLoop();
}
```

3.1.3 Funkcja RenderScene

W funkcji pojawia się reakcja na wciśnięty przycisk myszy w postaci modyfikacji kąta obrotu o kąt proporcjonalny do różnicy położeń kursora myszy. Dodatkowo ograniczono przybliżenie i oddalenie obrazu w celu czytelności wyniku.

```
// Funkcja określająca co ma być rysowane (zawsze wywoływana, gdy trzeba
// przerysować scenę)
void RenderScene(void)
{

   // Czyszczenie buforów koloru i głębokości
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   // Czyszczenie macierzy bieżącej
   glLoadIdentity();

   // Zdefiniowanie położenia obserwatora
   gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

   // Jeśli lewy klawisz myszy wciśnięty
   if (status == 1)
   {
      // Modyfikacja kąta obrotu o kat proporcjonalny
     // do różnicy położeń kursora myszy
      theta[0] += delta_x * pix2anglex;
```

```
        theta[1] += delta_y * pix2angley;
    }

    // Jeśli prawy klawisz myszy wciśnięty
    if (status == 2)
    {
        //Modyfikacja współrzędnej obserwatora o kat proporcjonalny
       // do różnicy położeń kursora myszy
        viewer[2] += delta_y * pix2angley;
        // Granica przybliżenia i oddalenia
        if (viewer[2] < 1.0f) viewer[2] = 1.01f;
        if (viewer[2] > 30.0f) viewer[2] = 29.99f;
    }

    //Obrót obiektu o nowy kąt
    glRotatef(theta[0], 0.0f, 1.0f, 0.0f);

    glRotatef(theta[1], 1.0f, 0.0f, 0.0f);

    // Ustawienie koloru rysowania na biały
    glColor3f(1.0f, 1.0f, 1.0f);

    // Narysowanie czajnika
    glutWireTeapot(3.0f);

    // Przekazanie poleceń rysujących do wykonania
    glutSwapBuffers();
}
```

3.1.4 Funkcja Mouse

```
//Funkcja sprawdza stan myszy i ustawia wartości zmiennych globalnych
void Mouse(int btn, int state, int x, int y)
{
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        //Przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
        x_pos_old = x;

        //Wciśnięty został lewy klawisz myszy
        status = 1;
    }
    else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        //Przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
        y_pos_old = y;

        //Wciśnięty został prawy klawisz myszy
        status = 2;
    }
    //Nie został wciśnięty żaden klawisz
    else status = 0;
}
```

3.1.5 Funkcja Motion

```
//Funkcja sprawdza położenie kursora myszy i ustawia wartości zmiennych globalnych
void Motion(GLsizei x, GLsizei y)
{
```

```
    //Obliczenie różnicy położenia kursora myszy
    delta_x = x - x_pos_old;

    delta_y = y - y_pos_old;

    //Podstawienie bieżącego położenia jako poprzednie
    x_pos_old = x;

    y_pos_old = y;

    //Przerysowanie obrazu sceny
    glutPostRedisplay();
}
```

3.1.6 Funkcja ChangeSize

Funkcja ChangeSize odpowiedzialna za utrzymanie stałych proporcji ekranu posiada dwie nowe zmienne:

```
// przeliczenie pikseli na stopnie
pix2anglex = 360.0 / (float)horizontal;
pix2angley = 360.0 / (float)vertical;
```

Wykorzystywane są one do przeliczenia pikseli szerokości oraz wysokości na stopnie tak aby niezależnie od proporcji okna obiekt reagował tak samo na ruch myszą.

Dodatkowo użyto funkcji gluPerspective, której założenie tłumaczy rysunek 1. Funkcja służy do definiowania obszaru widoczności dla rzutu perspektywicznego.

```
// Ustawienie parametrów dla rzutu perspektywicznego
gluPerspective(70, 1.0f, 1.0f, 30.0f);
```

3.2 Zadanie drugie

Poniżej zostały opisane elementy programu które nie pojawiły się w poprzednim zadaniu lub w poprzednich ćwiczeniach.

3.2.1 Biblioteki i zmienne globalne

```
//Promień sfery powierzchni, na której znajduje się obserwator
GLfloat r = 10.0f;

//Azymut, na której znajduje się obserwator
GLfloat azimuth = 0.0f;

//Elewacja, na której znajduje się obserwator
GLfloat elevation = 0.0f;
```

3.2.2 Funkcja RenderScene

```
// Jeśli lewy klawisz myszy wciśnięty
    if (status == 1)
    {
        //Modyfikacja azymutu i elewacji o kąt proporcjonalny
        //do różnicy położeń kursora myszy
        azimuth += delta_x * pix2anglex * 0.5f;
        elevation += delta_y * pix2angley * 0.5f;
```

```
    //Obliczenie współrzędnych obserwatora
    viewer[0] = getX(r, azimuth, elevation);
    viewer[1] = getY(r, azimuth, elevation);
    viewer[2] = getZ(r, azimuth, elevation);
  }

  // Jeśli prawy klawisz myszy wciśnięty
  if (status == 2)
  {
    //Modyfikacja promienia o kąt proporcjonalny
    //do różnicy położeń kursora myszy
    r += delta_y * pix2angley;
    //Ograniczenie przybliżenia i oddalenia
    if (r < 3.0f) r = 3.0f;
    if (r > 20.0f) r = 19.99f;

    //Obliczenie współrzędnych obserwatora
    viewer[0] = getX(r, azimuth, elevation);
    viewer[1] = getY(r, azimuth, elevation);
    viewer[2] = getZ(r, azimuth, elevation);
  }
```

3.2.3 Funkcje getX, getY, getZ

```
//Funkcja zwraca współrzędną x, wiążącą położenie obserwatora z
//azymutem, kątem elewacji i promieniem sfery powierzchni, na której znajduje się obserwator
GLfloat getX(GLfloat r, GLfloat azimuth, GLfloat elevation)
{
  return r * cosf(azimuth * M_PI / 180) * cosf(elevation * M_PI / 180);
}

//Funkcja zwraca współrzędną y, wiążącą położenie obserwatora z
//azymutem, kątem elewacji i promieniem sfery powierzchni, na której znajduje się obserwator
GLfloat getY(GLfloat r, GLfloat azimuth, GLfloat elevation)
{
  return r*sinf(elevation * M_PI / 180);
}

//Funkcja zwraca współrzędną z, wiążącą położenie obserwatora z
//azymutem, kątem elewacji i promieniem sfery powierzchni, na której znajduje się obserwator
GLfloat getZ(GLfloat r, GLfloat azimuth, GLfloat elevation)
{
  return r * sinf(azimuth * M_PI / 180) * cosf(elevation * M_PI / 180);
}
```

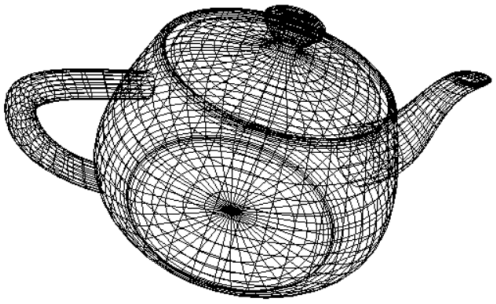
4. Wyniki

Kolory zrzutów ekranu zostały odwrócone w celu oszczędzenia tuszu, zgodnie z poleceniem prowadzącego.

4.1 Zadanie pierwsze

```
W celu modyfiakcji polozenia czajnika nalezy uzyc:
LPM + ruch myszy - Obracanie obiektu
PPM + ruch myszy - Przyblizanie obiektu
```

Rysunek 1 – Przykładowe działanie programu w oknie konsoli



Rysunek 2 – Wyświetlenie obrazu czajnika po obróceniu myszką

4.2 Zadanie drugie

```
W celu zmiany trybu wyswietlania obrazu jajka nalezy nacisnac:
'p' - siatka punktow
'w' - siatka punktow polaczona liniami
's' - siatka punktow polaczona w trojkaty

W celu modyfiakcji polozenia obserwatora nalezy uzyc:
LPM + ruch myszy - Obracanie obiektu
PPM + ruch myszy - Przyblizanie obiektu
```

Rysunek 3 – Przykładowe działanie programu w oknie konsoli

Rysunek 4 – Wyświetlenie obrazu jajka o wciśnięciu klawisza „s” po obróceniu myszką

5. Wnioski

Dzięki dokumentacji oraz instrukcji ze strony prowadzącego byłem w stanie stworzyć symulację obracającą czajnik oraz jajko. Symulacja zwróciła oczekiwane wyniki, program został wykonany poprawnie.

6. Źródła

Instrukcja do ćwiczenia ze stronu kursu:
http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/gk/lab/cw_4_dz/

Dokumentacje OpenGL i GLUT:
https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/
https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Źródła były dostępne w dniu wykonywana ćwiczenia.