Kurs:
Grafika komputerowa i komunikacja człowiek-komputer - Laboratorium

OpenGL - interakcja z użytkownikiem

Autor:
Aleksandra Chrustek, 263900

Prowadzący:
dr inż. Jan Nikodem

12.12.2023

1 Wstęp

Celem trzeciego laboratorium było zapoznanie się z zagadnieniem interakcji obiektów graficznych z użytkownikiem. Do zrozumienia tematu niezbędne było zapoznanie się z nowymi funkcjami bibliotek OpenGL i GLUT, jak również z instrukcją zamieszczoną na stronie ZSK, która pozwalała na osiągnięcie wiedzy postawowej i niezbędnej, aby przystąpienie do ćwiczenia było możliwe.

2 Przebieg laboratorium

Pierwszym uruchomionym programem był czajnik w rzucie ortograficznym, stworzony na podstawie kodu przedstawionego w instrukcji. Następnie wygenerowany został on w rzucie perspektywicznym, co pozwoliło na porównanie tych dwóch rzutowań i wybranie tego, na którym obiekt jest bardziej zbliżony do swojego trójwymiarowego pierwowzoru (rzutowanie perspektywiczne). Kolejną częścią laboratorium było zadanie do samodzielnego zrealizowania - wprawienie czajnika w ruch, tak aby ruszając myszką w lewo i prawo obracał się wokół osi Y.

3 Realizacja zadania

3.1 Obrót czajnika

Pierwszym zadaniem było wprawienie czajnika w ruch, tak aby ruszając myszką w poziomie kręcił się wokół osi Y, a w pionie wokół osi X. Obiekt miał również składać te dwa ruchy, aby poruszać się w pełnym zakresie. Miała także zostać zaimplementowana opcja przybliżania (należało zablokować możliwość „wchodzenia" w obiekt). W tym przypadku obiekt jest poruszany, a obserwator stoi w miejscu. Z tego powodu oś Z jest niewidoczna na animacji.

3.1.1 Funkcja Mouse(int btn, int state, int x, int y)

Funkcja Mouse(int btn, int state, int x, int y) obsługuje zdarzenia myszy, reagując na naciśnięcia i zwolnienia przycisków myszy.

Parametry:
-btn: Przechowuje informację o przycisku myszy, który został naciśnięty lub zwolniony (np. GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON).
-state: Przechowuje informację o stanie przycisku myszy - czy został naciśnięty (GLUT_DOWN) czy zwolniony (GLUT_UP).
-x i y: Przechowują aktualne położenie kursora myszy w oknie.

Zadania funkcji:
-Naciśnięcie lewego przycisku myszy (GLUT_LEFT_BUTTON): Zapisuje aktualne położenie kursora myszy (x_pos_old, y_pos_old). Ustawia stan przycisku myszy na 1 (status = 1).
-Naciśnięcie prawego przycisku myszy (GLUT_RIGHT_BUTTON): Zapisuje aktualne położenie kursora myszy (z_pos_old). Ustawia stan przycisku myszy na 2 (status = 2).
-Inne przypadki (np. zwolnienie przycisku): Ustawia stan przycisku myszy na 0 (status = 0).

```
void Mouse(int btn, int state, int x, int y)
{
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
x_pos_old = x;        // przypisanie aktualnie odczytanej pozycji kursora jako pozycji
  poprzedniej
y_pos_old = y;
status = 1;           // wcisniety zostal lewy klawisz myszy
}

else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
{
z_pos_old = y;
status = 2;  // wciśnięty został prawy klawisz myszy
}

else
status = 0;         // nie został wciśnięty żaden klawisz
}
```

3.1.2 Motion(GLsizei x, GLsizei y)

Motion(GLsizei x, GLsizei y) monitoruje położenie kursora myszy i dostosowuje wartości zmiennych globalnych związanych z ruchem myszy.

Zadania funkcji:
-Obliczenie różnicy położenia kursora myszy.
-Modyfikacja kątów obrotu (theta i theta2) w przypadku wciśnięcia lewego klawisza myszy (status == 1).
-Przybliżanie widoku w przypadku wciśnięcia prawego klawisza myszy (status == 2):

viewer[2] += delta_z; Sprawdzenie, czy nowa pozycja kamery mieści się w określonych granicach (od 0 do 360 stopni).
-Przerysowanie obrazu sceny.

```
void Motion(GLsizei x, GLsizei y)
{
delta_x = x - x_pos_old;     // obliczenie różnicy położenia kursora myszy
delta_y = y - y_pos_old;     // obliczenie różnicy położenia kursora myszy

if (status == 1)
{
theta += delta_x * pix2angle;  // Modyfikacja kąta obrotu o kąt
  proporcjonalny dla x do różnicy położenia kursora myszy
theta2 += delta_y * pix2angle; // Modyfikacja kąta obrotu o kąt
  proporcjonalny dla y do różnicy położenia kursora myszy
}
double pomocnicza = y - z_pos_old; // Zmienna pomocnicza do ustalania
 różnicy położenia kursora myszy przy przybliżaniu
if (status == 2 && viewer[2] + pomocnicza < 360 && viewer[2] + pomocnicza > 0)
{
viewer[2] += pomocnicza;
x_pos_old = x; // Ustawienie bieżącego położenia
  jako poprzednie
y_pos_old = y; // Ustawienie bieżącego położenia
  jako poprzednie
z_pos_old = y; // Ustawienie bieżącego położenia
  jako poprzednie
}

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

3.1.3 RenderScene()

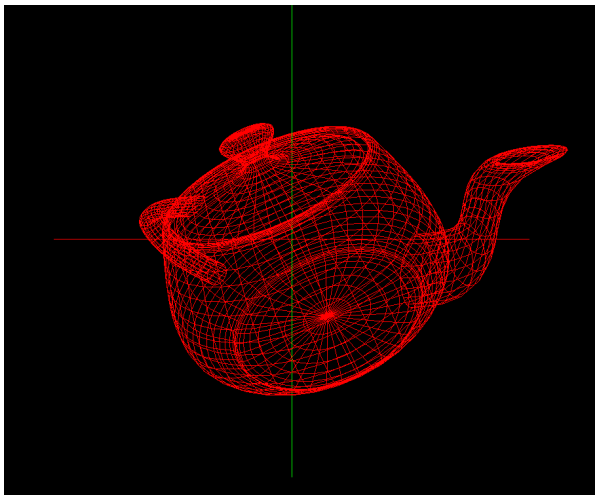
Funkcja RenderScene jest odpowiedzialna za renderowanie sceny, czyli rysowanie obiektów, osi, oraz zarządzanie ustawieniami kamery.

Zadania funkcji:
-Czyszczenie bufora: glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
-Ustawienie macierzy na jednostkową i zdefiniowanie położenia obserwatora: glLoadI-

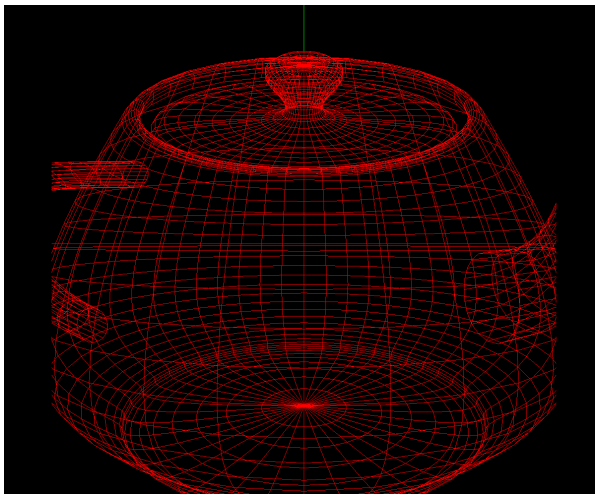
dentity(); gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
-Rysowanie osi współrzędnych (Axes()): Wywołuje funkcję Axes(), która rysuje trzy osie układu współrzędnych.
-Ustawienie koloru obiektu w zależności od wartości zmiennej kolor.
-Modyfikacja kątów obrotu (theta i theta2) w przypadku wciśnięcia lewego klawisza myszy (status == 1).
-Obrót obiektu zgodnie z wartościami kątów obrotu: glRotatef(theta, 0.0, 1.0, 0.0); glRotatef(theta2, 1.0, 0.0, 0.0);.
-Rysowanie czajnika (glutWireTeapot(3.0)).
-Renderowanie (glFlush() i glutSwapBuffers()): glFlush() przesyła polecenia do wykonania. glutSwapBuffers() zamienia przedni bufor z tylnym, co pozwala na płynne przejścia między klatkami.

```
void RenderScene(void)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Czyszczenie okna aktualnym kolorem czyszczącym
glLoadIdentity();
// Czyszczenie macierzy bieżącej - zamienienie jej na macierz jednostkową
gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
// Zdefiniowanie położenia obserwatora
Axes();
// Narysowanie osi przy pomocy funkcji zdefiniowanej wyżej

if (kolor == 1) glColor3f(0.0f, 1.0f, 0.0f);
else if (kolor == 2) glColor3f(1.0f, 0.0f, 0.0f);
else if (kolor == 3) glColor3f(0.0f, 0.0f, 1.0f);
// Ustawienie koloru na wybrany przez użytkownika

if (status == 1)                    // jeśli lewy klawisz myszy wciśnięty
{
theta += delta_x * pix2angle;    // modyfikacja kąta obrotu o kat
  proporcjonalny do różnicy położeń kursora myszy
}

glRotatef(theta, 0.0, 1.0, 0.0); //obrót obiektu o nowy kąt
glRotatef(theta2, 1.0, 0.0, 0.0); //obrót obiektu o nowy kąt

glutWireTeapot(3.0);
// Narysowanie czajnika
```

```
glFlush();
// Przekazanie poleceń rysujących do wykonania
glutSwapBuffers();
// Zamiana przedniego bufora (gdzie aktualnie są przechowywane dane) na bufor
 tylny - zawartość bufora tylnego będzie niezdefiniowana
}
```



Rysunek 1: Czajnik zdj.1



Rysunek 2: Czajnik zdj.2

3.2 Obrót jajka

Kolejne zadanie polegało na poruszeniu jajka z poprzedniego laboratorium. W tym przypadku to obiekt miał być statyczny, a poruszać miał się obserwator. Dzięki temu możliwe było zaobserwowanie ruszającej się osi Z.

3.2.1 Funkcja Mouse(int btn, int state, int x, int y)

Funkcja Mouse(int btn, int state, int x, int y) obsługuje zdarzenia myszy, reagując na naciśnięcia i zwolnienia przycisków myszy.

Parametry:
-btn: Przechowuje informację o przycisku myszy, który został naciśnięty lub zwolniony (np. GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON).
-state: Przechowuje informację o stanie przycisku myszy - czy został naciśnięty (GLUT_DOWN) czy zwolniony (GLUT_UP).
-x i y: Przechowują aktualne położenie kursora myszy w oknie.

Zadania funkcji:
-Naciśnięcie lewego przycisku myszy (GLUT_LEFT_BUTTON): Zapisuje aktualne położenie kursora myszy (x_pos_old, y_pos_old). Ustawia stan przycisku myszy na 1 (status = 1).
-Naciśnięcie prawego przycisku myszy (GLUT_RIGHT_BUTTON): Zapisuje aktualne położenie kursora myszy (z_pos_old). Ustawia stan przycisku myszy na 2 (status = 2).
-Inne przypadki (np. zwolnienie przycisku): Ustawia stan przycisku myszy na 0 (status = 0).

```
void Mouse(int btn, int state, int x, int y)
{
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
x_pos_old = x;        // przypisanie aktualnie odczytanej pozycji kursora jako pozycji
  poprzedniej
y_pos_old = y;
status = 1;           // wcisniety zostal lewy klawisz myszy
}

else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
{
```

```
z_pos_old = y;
status = 2;  // wciśnięty został prawy klawisz myszy
}

else
status = 0;         // nie został wciśnięty żaden klawisz
}
```

3.2.2 Funkcja Motion(GLsizei x, GLsizei y)

Funkcja Motion(GLsizei x, GLsizei y) monitoruje ruch myszy i aktualizuje wartości zmiennych globalnych, co umożliwia interakcję z programem w zależności od ruchu myszy.

Parametry:
-x i y: Aktualne położenie kursora myszy w oknie.

Zadania funkcji:
-Obliczanie różnicy pomiędzy aktualną a poprzednią pozycją kursora myszy.
-Jeśli lewy przycisk myszy jest naciśnięty (status == 1): Modyfikuje kąty obrotu obiektu (theta i fi) na podstawie ruchu myszy. Ogranicza kąty, aby nie przekroczyć 360 stopni.
-Jeśli prawy przycisk myszy jest naciśnięty (status == 2): Modyfikuje promień oddalania/przybliżania obserwatora (R) na podstawie ruchu myszy w osi Y. Ogranicza wartość promienia, aby nie wyszedł poza ustalone zakresy.
-Aktualizacja pozycji kursora myszy.
-Przerysowanie sceny.

```
void Motion(GLsizei x, GLsizei y)
{
delta_x = x - x_pos_old; // obliczenie różnicy położenia kursora myszy
delta_y = y - y_pos_old; // obliczenie różnicy położenia kursora myszy

if (status == 1)  // Jeśli lewy klawisz wciśnięty
{
theta += delta_x * pix2angle_x; // modyfikacja kąta obrotu
fi += delta_y * pix2angle_y; // do róznicy połozenia kursora myszy
if (theta >= 360.0) // jeśli kąt >= 360 stopni
theta = 0.0; // to kąt = 0
if (fi >= 360.0)
```

```
fi = 0.0;
}
else if (status == 2) {  // Jeśli lewy klawisz wciśnięty
R += 0.01 * delta_y; // przyblizanie się obserwatora do obiektu
if (R <= 8.0)  // ustalone maksymalne przyblizenia i oddalenia
R = 8.0;  // aby nie wejść w środek jajka
if (R >= 13.0)
R = 13.0;
}
x_pos_old = x;           // ustawienie aktualnego położenia jako poprzednie
y_pos_old = y;
z_pos_old = y;
glutPostRedisplay();    // przerysowanie obrazu sceny
}
```

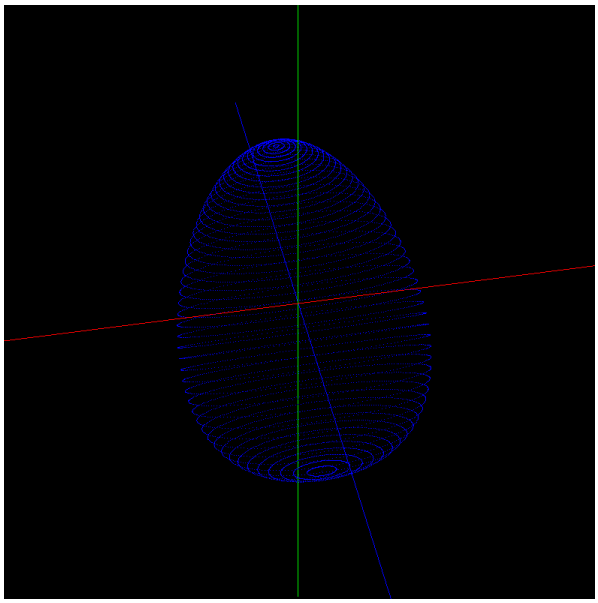
3.2.3 RenderScene()

Funkcja RenderScene jest odpowiedzialna za renderowanie sceny, czyli rysowanie obiektów, osi, oraz zarządzanie ustawieniami kamery.

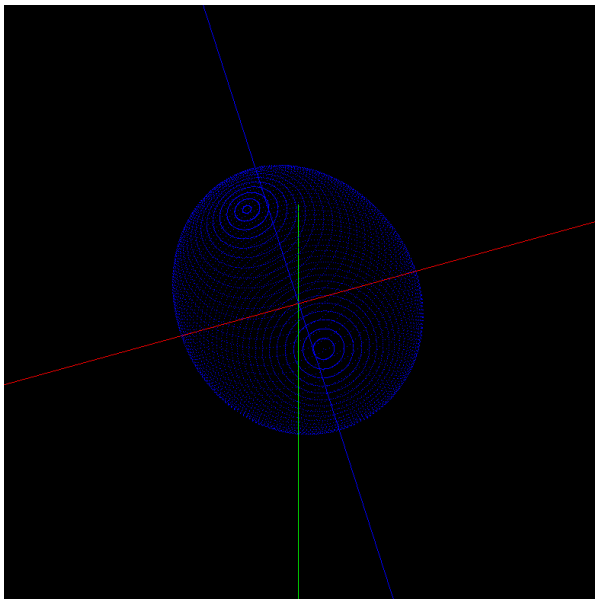
Zadania funkcji:
-Czyszczenie bufora: glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
-Ustawienie pozycji kamery (glLoadIdentity() i gluLookAt()): Resetuje macierz modelview (glLoadIdentity()). Ustawia pozycję kamery za pomocą gluLookAt(). Pozycja kamery jest zależna od wartości zmiennych viewer, theta, fi, R, angle.
-Rysowanie osi współrzędnych (Axes()): Wywołuje funkcję Axes(), która rysuje trzy osie układu współrzędnych.
-Rysowanie obiektu 3D (Egg()): Obraca obiekt zgodnie z wartościami zmiennych angle. Wywołuje funkcję Egg(), która rysuje trójwymiarowe jajko w zależności od wybranego modelu.
-Renderowanie (glFlush() i glutSwapBuffers()): glFlush() przesyła polecenia do wykonania. glutSwapBuffers() zamienia przedni bufor z tylnym, co pozwala na płynne przejścia między klatkami.

```
void RenderScene(void)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Czyszczenie okna aktualnym kolorem czyszczacym
glLoadIdentity();
// Czyszczenie macierzy biezacej- zamienienie jej na macierz jednostkową
```

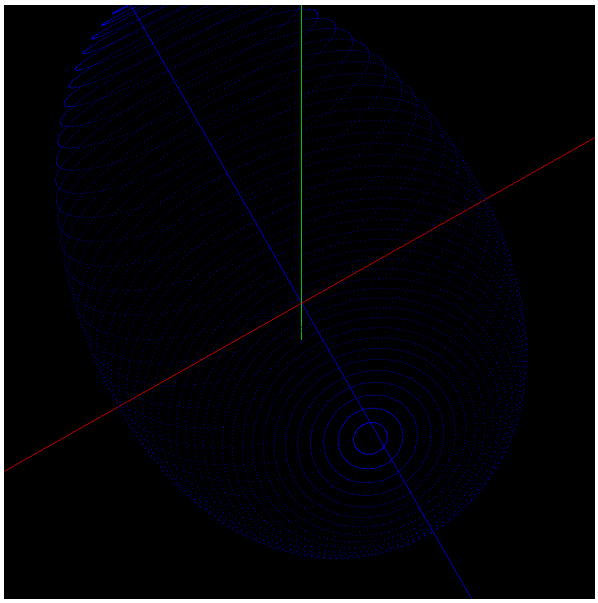
```
viewer[0] = R * cos(theta) * cos(fi);
viewer[1] = R * sin(fi);
viewer[2] = R * sin(theta) * cos(fi);

gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, cos(fi), 0.0);

glRotatef(theta, 0.0, 1.0, 0.0);  //obrót obiektu o nowy kat
glRotatef(theta2, 1.0, 0.0, 0.0);

Axes();
// Narysowanie osi przy pomocy funkcji zdefiniowanej wyzej
glColor3f(0.0f, 0.7f, 0.5f);
// Ustawienie koloru rysowania
glRotatef(angle[0], 1.0, 0.0, 0.0);
glRotatef(angle[1], 0.0, 1.0, 0.0);
glRotatef(angle[2], 0.0, 0.0, 1.0);

glRotated(45.0, 1.0, 0.0, 0.0);

Egg();

glFlush();
// Przekazanie polecen rysujacych do wykonania
glutSwapBuffers();
// Zamiana przedniego bufora (gdzie aktualnie są przechowywane dane) na bufor tylny
 - zawartość bufora tylnego będzie niezdefiniowana
}
```



Rysunek 3: Jajko zdj.1



Rysunek 4: Jajko zdj.2



Rysunek 5: Jajko zdj.3

4 Wnioski

Zadanie pozwoliło na poznanie i zastosowanie metod obrotu trójwymiarowego modelu. Korzystanie z bibliotek OpenGL i GLUT okazało się skutecznym narzędziem do implementacji funkcji obrotu. Zaimplementowanie dwóch różnych metod obrotu pozwoliło zobaczyć, jak wpływają one na odbiór grafiki 3D. Każdy z tych trybów ma swoje zastosowanie w różnych kontekstach wizualnych. Pozwala to na lepsze zrozumienie struktury modelu oraz potencjalnych detali z różnych perspektyw. Teoretycznie, animacje w grafice 3D polegają na manipulacji transformacjami obiektów w trójwymiarowej przestrzeni. W praktyce animacje często są wykorzystywane do poprawy doświadczenia użytkownika oraz do eksploracji trójwymiarowych struktur. W rezultacie, zdobyta wiedza i umiejętności są cennym dodatkiem do mojej wiedzy programistycznej. Eksperymentowanie z różnymi aspektami grafiki komputerowej otwiera drzwi do dalszego zgłębiania tego fascynującego obszaru informatyki.