POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 4

Interakcja z użytkownikiem

Wykonał:	**Kamil Kamyszek**
Termin:	**PT/NP 11.00-14.00**
Data wykonania ćwiczenia:	**09.11.2018r.**
Data oddania sprawozdania:	**22.11.2018r.**
Ocena:	

Uwagi prowadzącego:

1 Wstęp

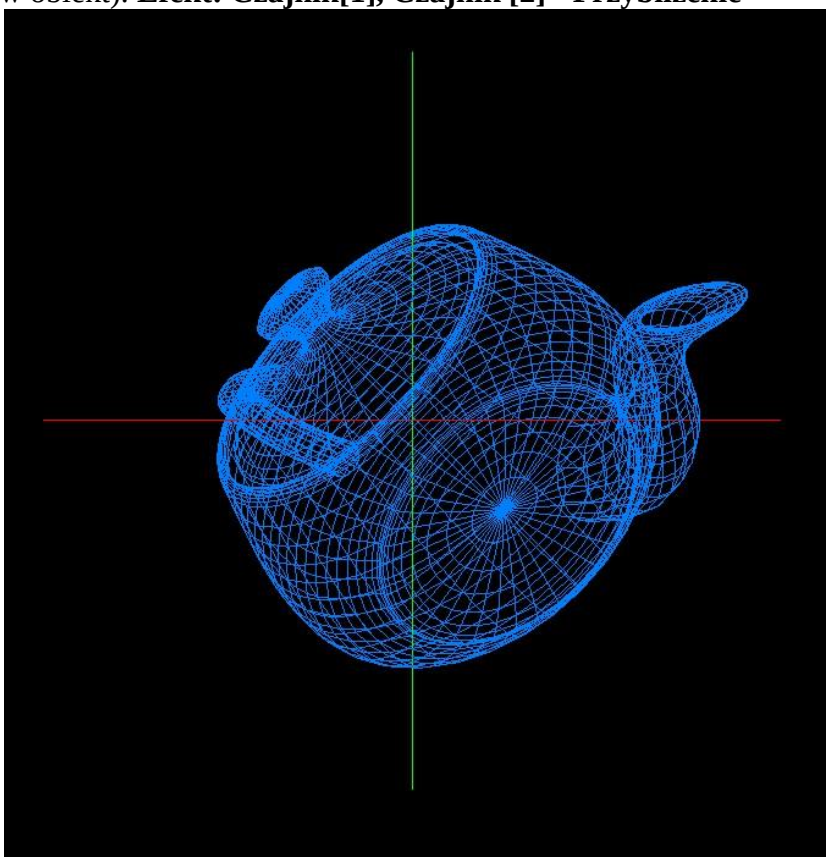
Na czwartym laboratorium z grafiki studenci mieli zaznajomić się zagadnieniem interakcji obiektów graficznych z użytkownikiem. Do zrozumienia tego tematu potrzebne było uważne przestudiowanie instrukcji zamieszczonej przez prowadzącego na stronie ZSK, jak również zaznajomienie się z nowo poznanymi funkcjami bibliotek OpenGL i GLUT.

2 Przebieg Laboratorium

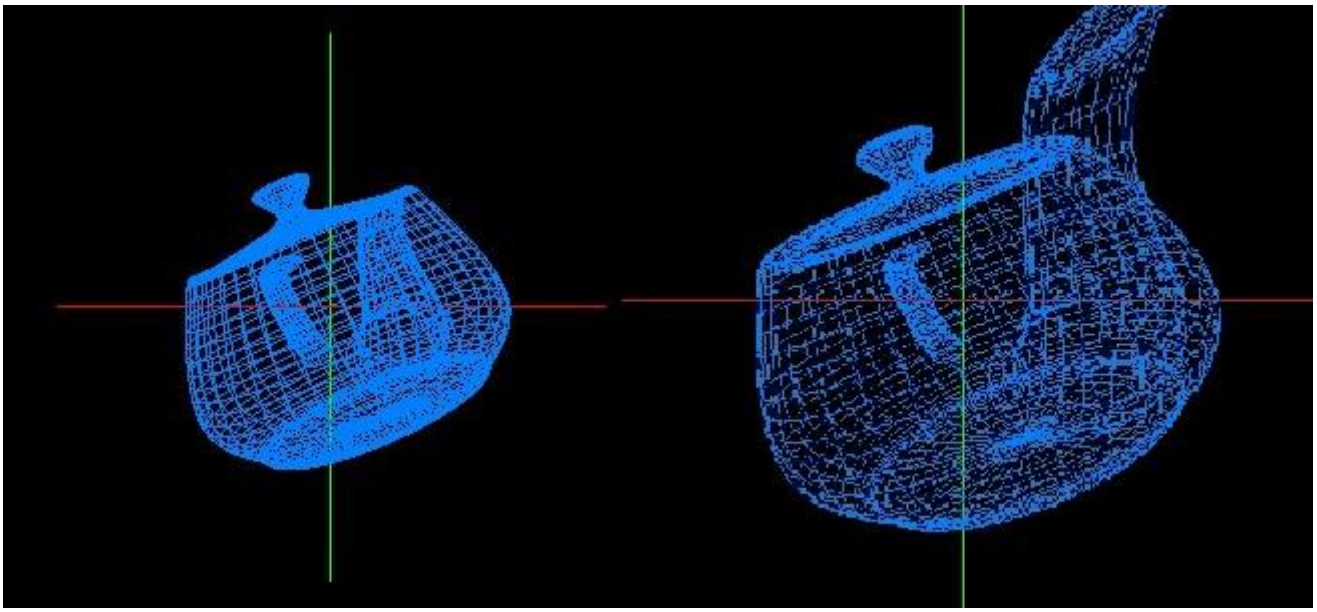
Na wstępie studenci mieli za zadanie wyświetlić czajnik w rzucie ortograficznym używając funkcji glOrtho(). Następnie wygenerowany został on w rzucie perspektywicznym co pozwoliło na porównanie tych dwóch rzutowań i wybranie tego, na którym obiekt jest bardziej zbliżony do swojego trójwymiarowego pierwowzoru (rzutowanie perspektywiczne). Ostatnim zadaniem przed przejściem do polecenia głównego było wprawienie w ruch czajnika tak aby ruszając myszką w lewo i prawo obracał się wokół osi Y.

3 Zadanie do samodzielnego wykonania

Jako pierwsze zadanie jakie studenci mieli wykonać samodzielnie 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 również zostać zaimplementowania opcja przybliżania (Trzeba było zablokować możliwość „wchodzenia" w obiekt). **Efekt: Czajnik[1], Czajnik [2] - Przybliżenie**



Rysunek 1 Czajnik[1]



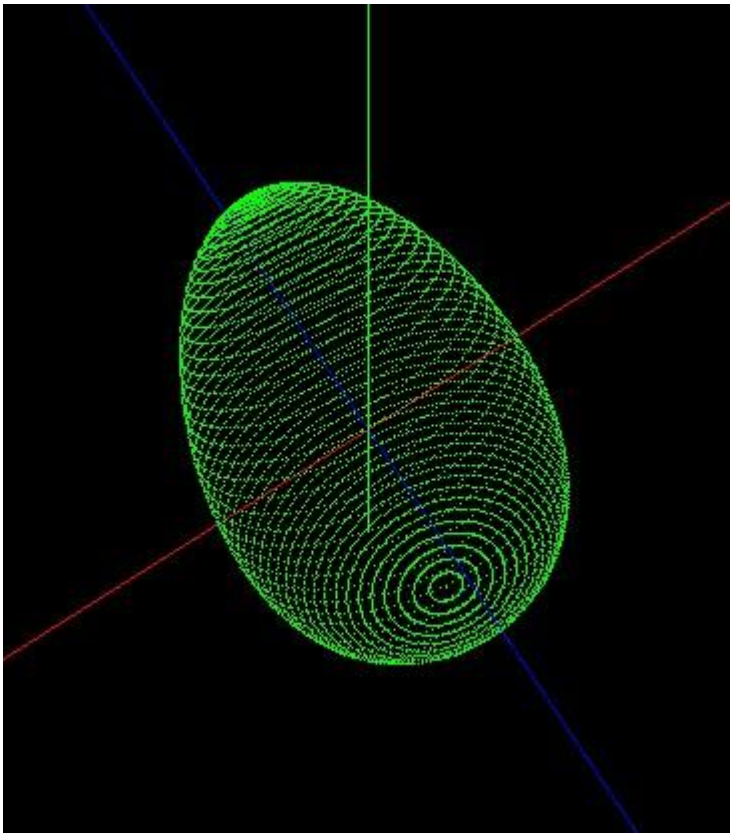
Rysunek 2 Czajnik [2] - Przybliżenie

Najważniejsze funkcje zawarte w programie:

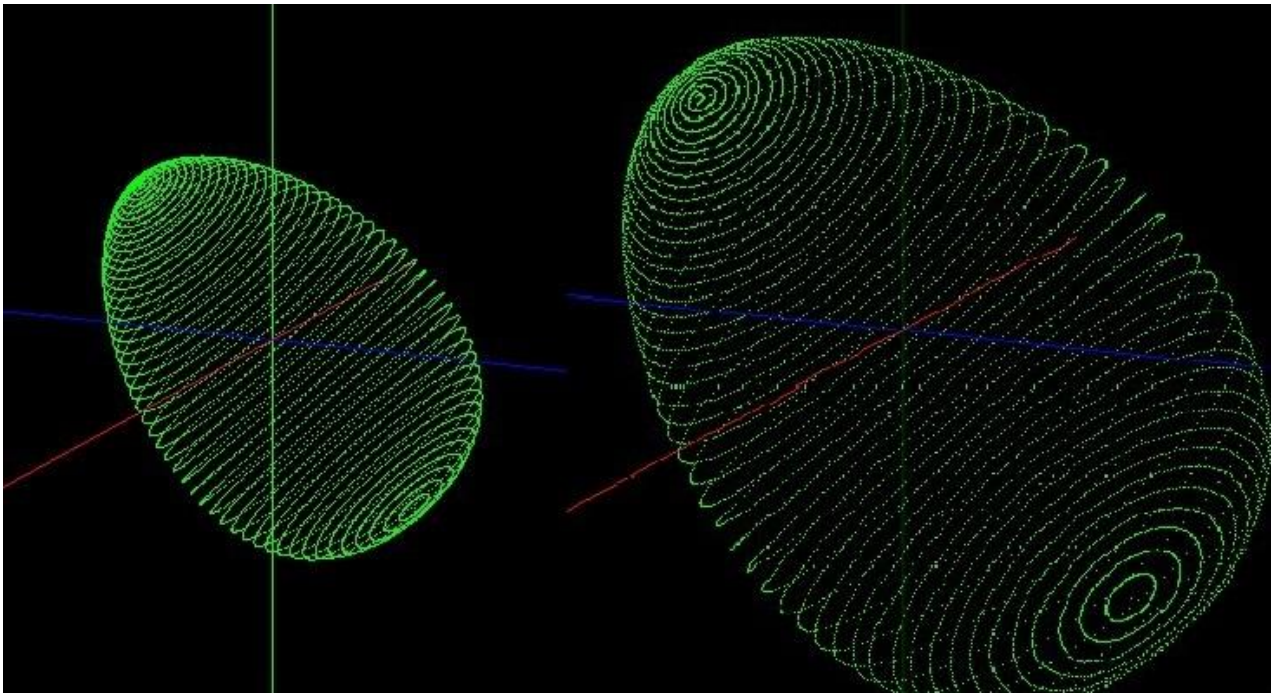
```
void Mouse(int btn, int state, int x, int y)//Obsługa myszy
{
	if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		x_pos_old = x;// przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
		y_pos_old = y;// przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
		status = 1;   //Wciśnięty lewy klawisz
	}
	else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		z_pos_old = y;// przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
		status = 2;   //Wciśnięty prawy klawisz
	}
	else
		status = 0;   //Nie wciśnięty żaden klawisz
}
```

```
void Motion(GLsizei x, GLsizei y)//Wprawienie obiektu w ruch
{
	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)		//Jeżeli lewy klawisz wciśnięty
	{	//modyfikacja kątów pobrana z instrukcji
		theta += delta_x * pix2angle;//modyfikacja kąta obrotu o kąt
proporcjonalny dla x do róznicy położenia kursora myszy
		theta2 += delta_y * pix2angle;//modyfikacja kąta obrotu o kąt
proporcjonalny dla y do róznicy położenia kursora myszy
	}
	double pom = y - z_pos_old; //Zmienna pomocnicza do ustalania różnicy położenia
kursora myszy przy przybliżaniu.
	//Jeśli prawy klawisz wciśnięty to można przybliżać w zakresie od 70 do 150.
if (status == 2 && viewer[2] + pom < 150 && viewer[2] + pom > 70) viewer[2] += pom;
	x_pos_old = x;			//Ustawienie bieżącego położenia jako poprzednie
	y_pos_old = y;
	z_pos_old = y;
	glutPostRedisplay();	// przerysowanie obrazu sceny
}
```

W tym przypadku to obiekt jest poruszany, a obserwator stoi w miejscu przez co oś ‘Z’ jest niewidoczna na animacji. W kolejnym zadaniu, które polegało na poruszeniu jajka z poprzedniego laboratorium to obiekt miał być statyczny a poruszać miał się obserwator, aby możliwe było zaobserwowanie ruszającej się osi ‘Z’. Oto **Efekty: Jajko[1], Jajko[2] - Przybliżenie**



Rysunek 3 Jajko[1]



Rysunek 4 Jajko[2] - Przybliżenie

W przypadku jajka funkcja odpowiadająca wykrywanie wciśniętych klawiszy myszy jest analogiczna do tego użytego w poprzednim zadaniu. Najważniejsze elementy kodu:

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

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

$$y_s(\Theta,\Phi) = R\sin(\Phi) \qquad 0 \le \Theta \le 2\pi \quad 0 \le \Phi \le 2\pi$$

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

Rysunek 5 Zależności wiążące położenie obserwatora z azymutem, katem elewacji i promieniem sfery na powierzchni, której znajduje się obserwator

Zależności zostały zapisane w funkcji RenderScene(). Oto one:

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

3 Wnioski

Dzięki zamieszczonej instrukcji na stronie zsk, wykonanie zadania nie sprawiło większych trudności i pozwoliło oswoić się z zagadnieniem interakcji z użytkownikiem. Można było to przetestować w przypadku obiektów wygenerowanych przez studentów na zajęciach. Biblioteki graficzne OpenGL i GLUT w dużym stopniu ułatwiają użytkownikowi zaprogramowanie ruchu, dzięki czemu można w krótkim czasie poruszyć obiekt, który się wykonało i obejrzeć z każdej strony np. w celu sprawdzenia czy wszystko wygenerowało się poprawnie i nie ma dziur w stworzonej strukturze. Nauczenie się poruszania obiektów statycznych ma bardzo wiele zastosowań w grafice 3D i jest niezbędne do jej tworzenia i testowania stworzonych struktur.