Sprawozdanie z ćwiczenia 4
OpenGL – interakcja z użytkownikiem
Maksymilian Iwanow
209946
Poniedziałek 10.15

Celem ćwiczenia było zrozumienie, jak przy pomocy funkcji biblioteki OpenGL z rozszerzeniem GLUT można zrealizować prostą interakcję, polegającą na sterowaniu ruchem obiektu i położeniem obserwatora w przestrzeni 3-D. Do sterowania służyła mysz.

Zadanie pierwsze polegało na wygenerowaniu trójwymiarowego obrazu czajnika w rzucie perspektywicznym i zapewnienie możliwości jego skalowania i obracania za pomocą myszy.

• wciśnięty lewy przycisk myszy i ruch w kierunku pionowym - obrót wokół osi y
• wciśnięty lewy przycisk myszy i ruch w kierunku poziomym - obrót wokół osi x
• wciśnięty prawy przycisk myszy i ruch w kierunku pionowym - zbliżenie bądź oddalenie

Funkcja void Mouse(int btn, int state, int x, int y) odpowiedzialna za odczytywanie sekwencji myszki:

```
void Mouse(int btn, int state, int x, int y)

{
	if ((btn == GLUT_RIGHT_BUTTON ) && (state == GLUT_DOWN))
	{
		statusp=1;
	}
	if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		x_pos_old=x;
		y_pos_old=y;
		status = 1;
	}
	else
		status = 0;

}
```

Funkcja void Motion(GLsizei x, GLsizei y) odpowiedzialna za odczytywanie sekwencji myszki wyznaczająca o ile stopni ma nastąpić obrót:

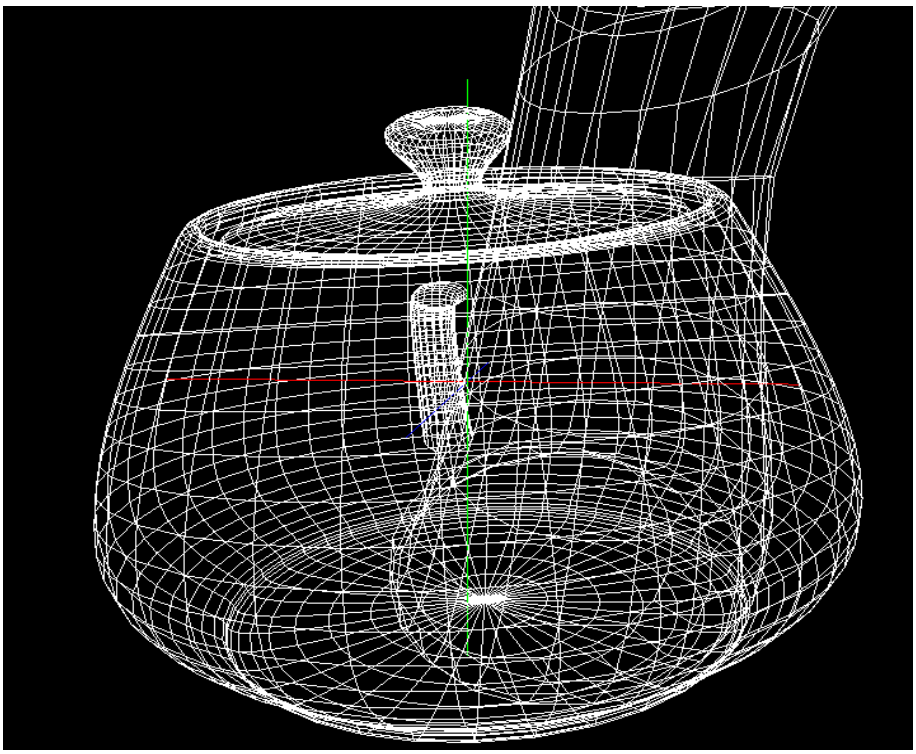
```
void Motion( GLsizei x, GLsizei y )
{

	delta_x=x-x_pos_old;
	x_pos_old=x;
	delta_y=y-y_pos_old;
	y_pos_old=y;
	glutPostRedisplay();

}
```

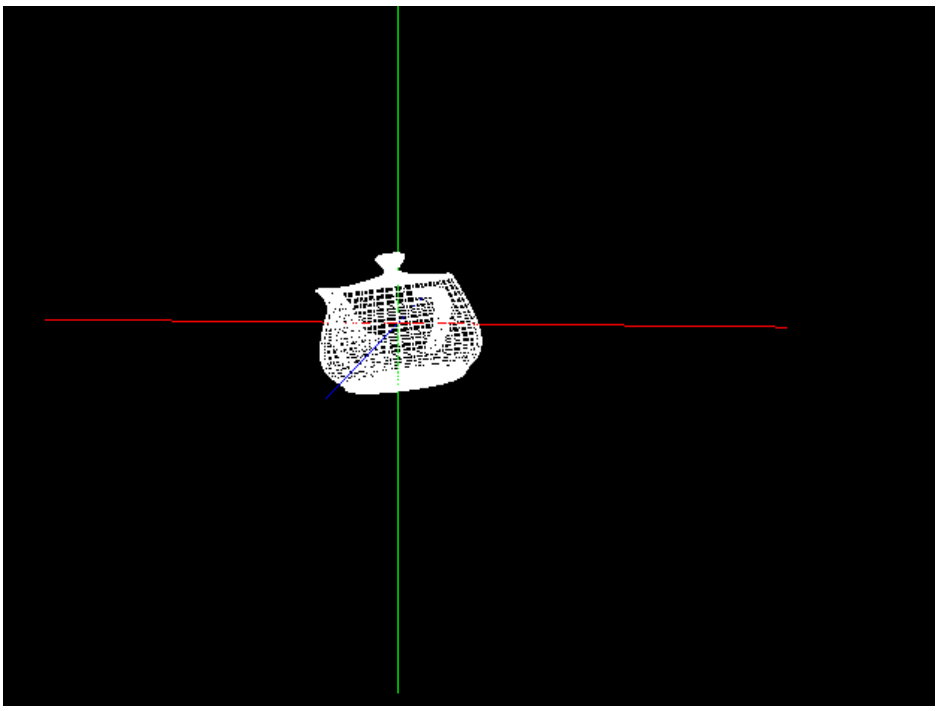
Funkcja odpowiedzialna za przybliżanie/oddalanie to `glScalef(zoom, zoom, zoom );`.

Funkcja void RenderScene(void) odpowiedzialna za renderowanie:

```
void RenderScene(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glutReshapeFunc(ChangeSize);
	glLoadIdentity();
	gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
	Axes();
	if(status == 1)
	{
		theta += delta_x*pix2angle/50;	// modyfikacja kąta obrotu o kat
		theta2 += delta_y*pix2angle/50;
	}
	else if (statusp==1)
	{
		if(delta_x>=delta_y) zoom=zoom+0.01;
		else zoom=zoom-0.01;

		if (zoom >=-0.01 && zoom <= 4.0) glScalef(zoom, zoom, zoom );
		else zoom = 1.0;

	}



	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
	glColor3f(1.0f, 1.0f, 1.0f);
	glutWireTeapot(3.0);
	glFlush();
	glutSwapBuffers();


}
```

Wynik działania programu:



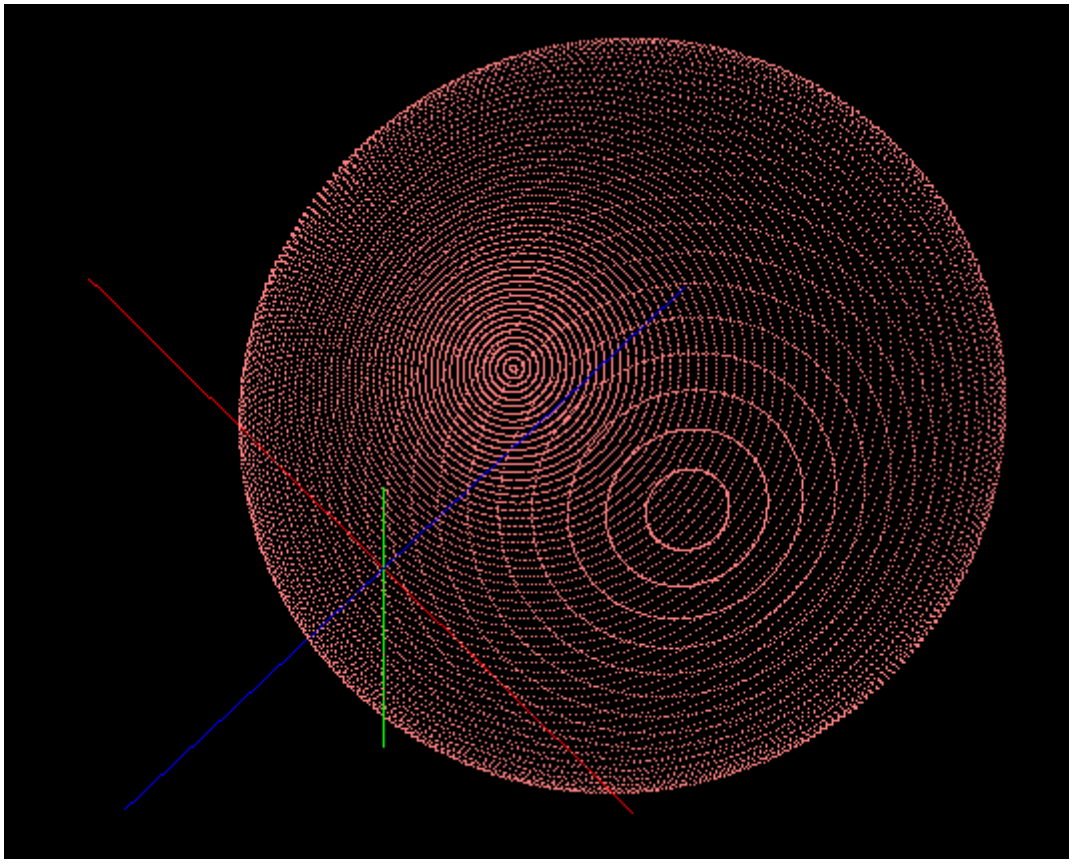


Zadanie drugie polegało na stworzeniu możliwości sterowania obiektem za pomocą zmiany punktu widzenia. Owym obiektem miało być jajko z poprzedniego laboratorium. Pierwszym założeniem było ustawienie jajka w środku układu. Obserwator może poruszać się po sferze kuli o zadanym promieniu.

Funkcja void RenderScene(void) odpowiedzialna za renderowanie:

```
void RenderScene(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glutReshapeFunc(ChangeSize);
	glLoadIdentity();
	gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
	Axes();
	if(status == 1)                     // jeśli lewy klawisz myszy wcięnięty
	{
		theta += delta_x*pix2angle/50;
		theta2 += delta_y*pix2angle/50;
	}
	else if (statusp==1)
	{
		if(delta_x>=delta_y) zoom=zoom+0.01;
		else zoom=zoom-0.01;
		if (zoom >=-0.01 && zoom <= 4.0) glScalef(zoom, zoom, zoom );
		else zoom = 1.0;

	}
	viewer[0]=Er*cos(theta)*cos(theta2);
	viewer[1]=Er*sin(theta2);
	viewer[2]=Er*sin(theta)*cos(theta2);
	glColor3f(1.0f, 1.0f, 1.0f);
	jajo1();
	glFlush();
	glutSwapBuffers();


}
```

Wynik działania programu:



Podsumowanie, wnioski, trudności:

1. Główną trudnością było zapisanie równań dla tablicy ' viewer ' określającej współrzędne obserwatora (x,y,z).
2. Kolejnym problemem było połączenie programów z dwóch laboratoriów w jeden oraz kontrola szybkości przy zmianie współrzędnych obserwatora.
3. Największą trudnością okazało się napisanie funkcji przybliżającej i oddalającej. Należało zabezpieczyć się przed „odbijaniem” obrazu przy za dużym przybliżeniu bądź oddaleniu.