Grafika komputerowa

Autor:
Jacek Wieczorek (181043)

Prowadzący:
Dr inż. Tomasz Kapłon

Wydział Elektroniki
III rok
Pn TP 08.15 - 11.00

28 listopada 2011

1 Cel laboratorium

Ćwiczenie ma za zadanie pokazać, 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 będzie mysz.

2 Zad 1

Celem pierwszego zadnia było wygenerowanie trójwymiarowego obrazu czajnika w rzucie perspektywicznym i zapewnienie możliwości jego skalowania i obracania za pomocą myszy :

- wciśnięty LPM i ruch w kierunku poziomym - obrót wokół osi y
- wciśnięty LPM i ruch w kierunku pionowym - obrót wokół osi x
- wciśnięty PPM i ruch w kierunku pionowym - zbliżenie (zoom)

Funkcja odpowiedzialna za odczytywanie odpowiednich sekwencji myszy :

```
1  void Mouse(int btn, int state, int x, int y)
   {
           if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
           {
                   x_pos_old = x;
                   y_pos_old = y;
                   status = 1;
           }
           else if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
           {
11                 zoom = y;
                   status = 2;
           }
           else
           {
                   status = 0;
           }
   }
```

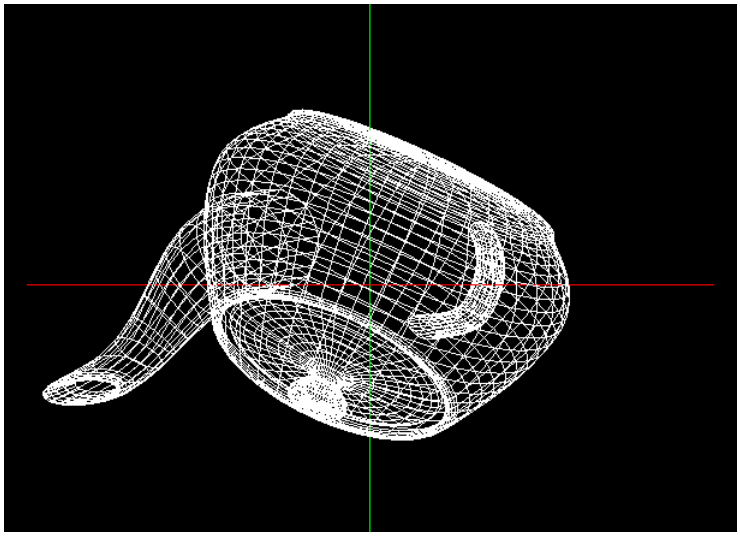
Funkcja odpowiedzialna za wyznaczanie o ile ma nastapić obrót lub zoom

```
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;
        delta_zoom = y - zoom;
        zoom = y;
        glutPostRedisplay();
}
```

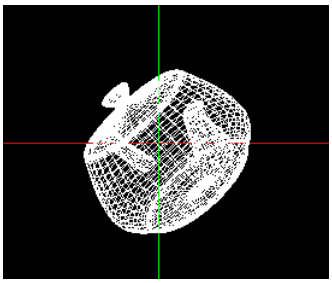
Funkcja odpowiedzialna za renderowanie widoku :

```
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)
        {
                thetax += delta_x * pix2angle;
                thetay += delta_y * pix2angle;
        }
        else if(status ==2)
        {
                theta_zoom += delta_zoom*pix2angle;
        }

        glRotatef(thetax, 0.0, 1.0, 0.0);
        glRotatef(thetay, 1.0, 0.0, 0.0);
        glRotatef(theta_zoom, 0.0, 0.0, 0.0);
        glColor3f(1.0f, 1.0f, 1.0f);

        glutWireTeapot(3.0);
        glFlush();
        glutSwapBuffers();
}
```

W celu zastosowania rzutu perspektywicznego, w funkcji *ChangeSize* zamiast funkcji *glOrtho*() skorzystaliśmy z funkcji *gluPerspective*().



Rysunek 1: Czajnik

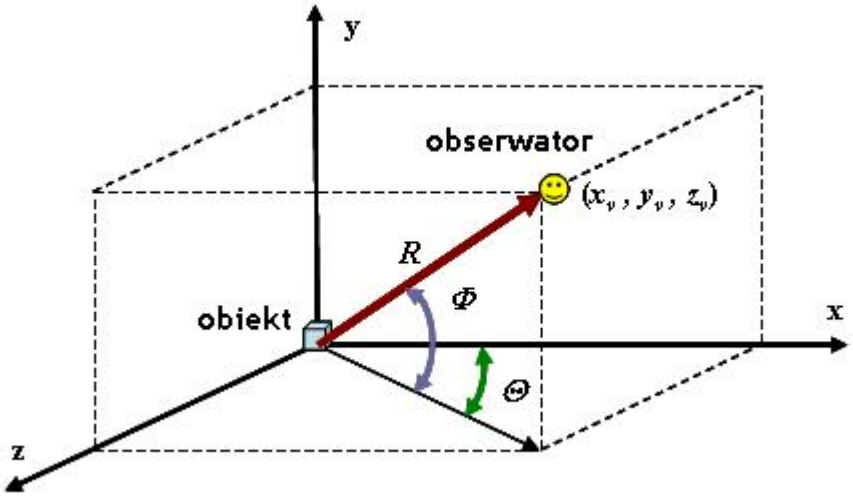


Rysunek 2: Czajnik - zoom

3 Zad 2

Drugie zadanie polegało na możliwości "sterowania" jajkiem z laboratorium 3 za pomocą zmiany punktu widzenia, przy następujących założeniach:

- Jajko znajduje się w środku układu współrzędnych
- Punkt obserwatora może się poruszać po po powierzchni sfery o promieniu R i środku będącym środkiem układu współrzędnych
- Sterowanie odbywać się będzie za pomocą dwóch kątów : kąt azymutu - Θ o kąt elewacji - Φ



Rysunek 3: Układ, obiekt, obserwator i kąty azymutu i elewacji

Zależności wiążące współrzędne punktu obserwatora z promieneim sfery i kątami:

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

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

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

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

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

Założenia sterowania:

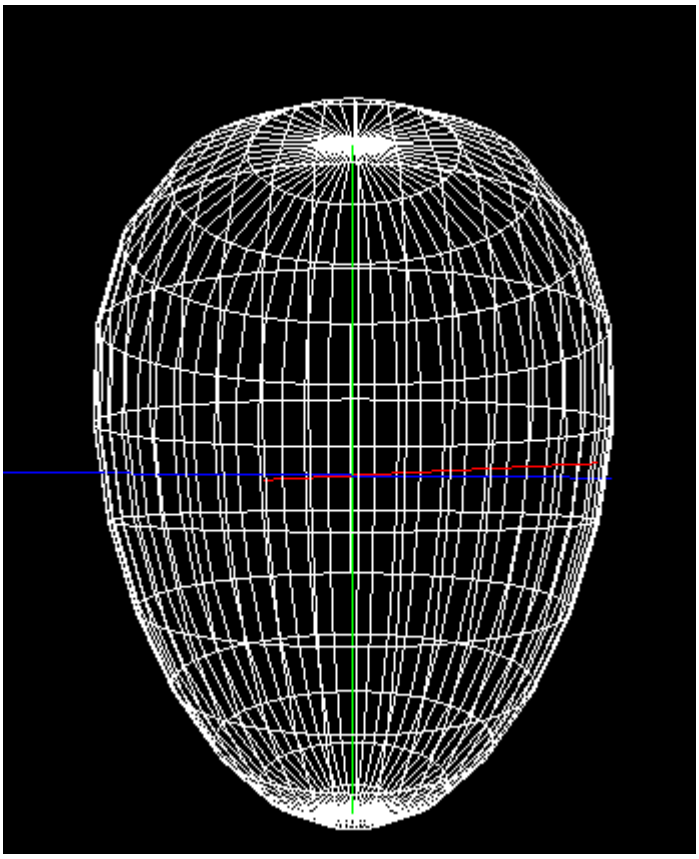
- Przy wciśniętym LPM i ruch w kierunku poziomym zmianie ulega kąt azymutu
- Przy wciśniętym LPM i ruch w kierunku pionowym zmianie ulega kąt elewacji
- Przy wciśniętym PPM i ruchu w kierunku pionowym ulega zmianie R

Funkcja odpowiedzialna za renderowanie widoku :

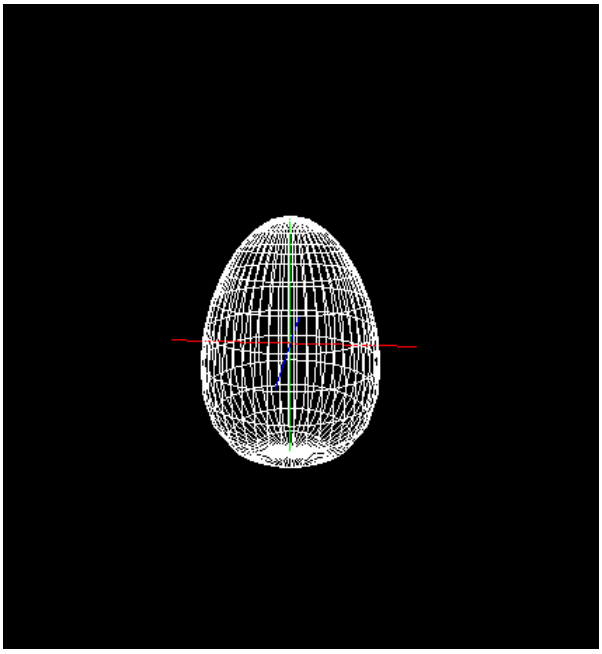
```
void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, p, 0.0)
        ;
    Axes();
    if(status == 1)
    {
       thetax += delta_x * pix2angle / 30.0;
       thetay += delta_y * pix2angle / 30.0;

    }  else if(status ==2){
            theta_zoom += delta_zoom/10.0;

    }

    if(thetay > 3.1415) thetay -= 2*3.1415;
    else if(thetay <= -3.1415) thetay += 2*3.1415;


    if(thetay > 3.1415/2 || thetay < -3.1415/2)
    {
            p = -1.0;
    } else
    {
            p = 1.0;
    }


    viewer[0] = theta_zoom*cos(thetax)*cos(thetay);
    viewer[1] = theta_zoom*sin(thetay);
    viewer[2] = theta_zoom*sin(thetax)*cos(thetay);

    if(model == 1)
            EggsPoints();
    else if(model == 2)
            EggsMesh();
    else
            EggsTriangles();

    glFlush();
    glutSwapBuffers();
}
```

W celu określenia położenia obserwatora wykorzystujemy funkcję $gluLoolAt()$. Trójelementowa tablica *viewer* określa nam współrzędne x, y, z obserwatora wyliczone za pomocą powyższych wzorów. Następne 3 parametry funkcji określają położenie środka układu współrzędnych, a ostatnie 3 kierunki wektorów. Ze względu na okresowość funkcji sinus i problem z pelnym obrotem jajka z powodu wartości zmiennej y, w określaniu kierunku wektora Y skorzystaliśmy z parmetru p, który przyjmuje odpowiednio wartości 1 i -1. W tym celu sprawdzamy czy kąt Φ nie przekracza wartości Π lub $-\Pi$, jeżeli tak to dodajemy lub odejmujemy od niego równowarość $2 * \Pi$, i w zalezności od wartości kąta Φ ustalamy odpowiednią wartość parametru p.

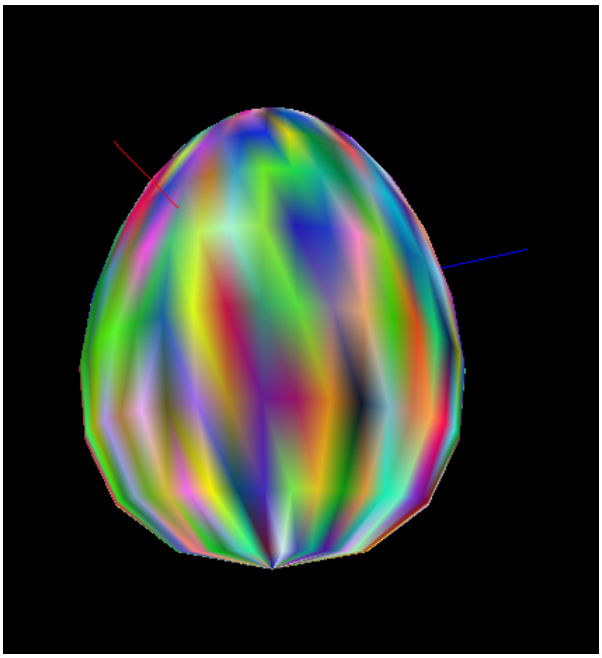
Pozostałe funkcje strujące są takie same jak w zadaniu poprzednim. W zadaniu wykorzystane zostało jajko z laboratorium nr 3.



Rysunek 4: Jajko "do góry nogami"



Rysunek 5: Jajko ze zmienjszonym promieniem R



Rysunek 6: Jajko zbudowane z trójkątów

4 Wnioski

Interakcja obiektów w przestrzeni $3D$ z użytkownikiem z wykorzystaniem biblioteki *OpenGL* i *GLUT* nie jest tak skomplikowanym zadaniem. Probelemem okazały się matematyczne opisy równań, a zwłaszcza wyeliminowanie błędu obrotu jajka wynikającego z okresowości funckji sinus. Sama obsługa zdarzeń pochodzących od klawiatury, czy też myszy jest prosta i intuicyjna.