Grafika komputerowa

Autor:
Tymon Tobolski (181037)

Prowadzący:
Dr inż. Tomasz Kapłon

Wydział Elektroniki
III rok
Pn TP 08.15 - 11.00

14 listopada 2011

1 Cel laboratorium

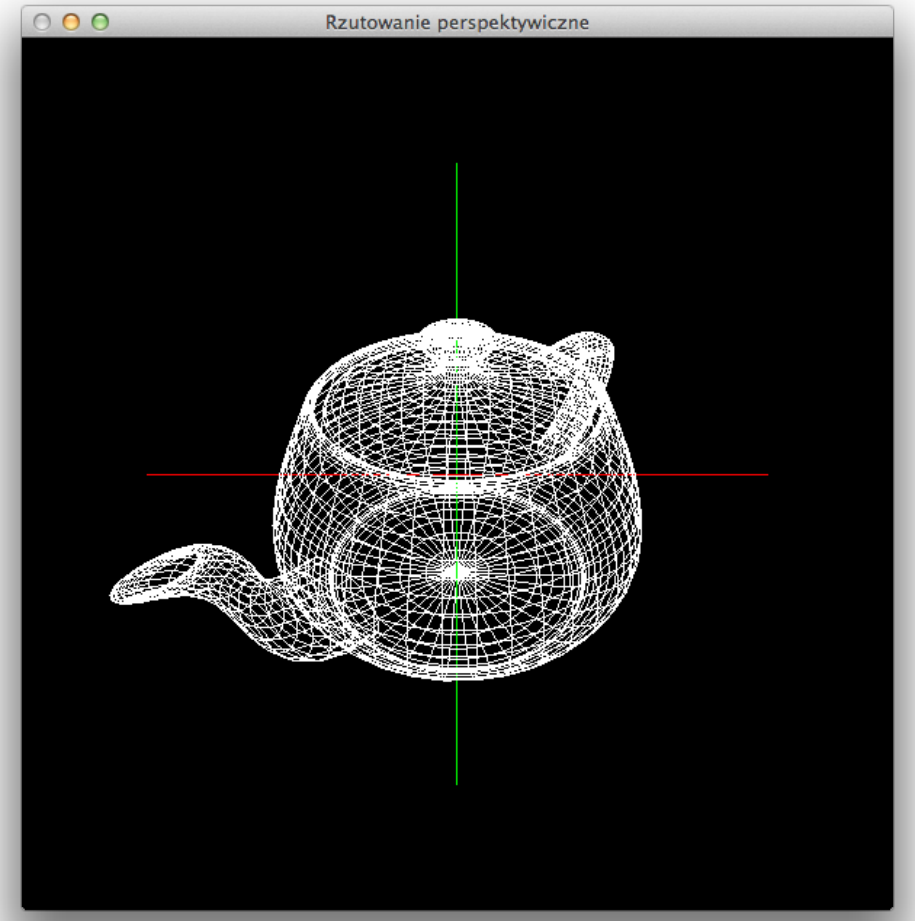
Celem laboratorium była prezentacja obiektów trójwymiarowych w rzucie perspektywicznym oraz realizacja prostej interakcji z użytkownikiem polegającej na sterowaniu ruchem obiektu i położeniem obserwatora w przestrzeni 3-D za pomocą myszy.

2 Obrót obiektu

Program wyświetla czajnik i pozwala na jego obrót za pomocą myszy. Po wciśnięciu lewego przycisku myszy i przesunięciu kursora w lewo lub w prawo czajnik obraca się wokół osi Y, natomiast przesunięcie kursora w górę lub w dół ekranu powoduje obrót obiektu wokół osi X. Ponadto wciśnięcie prawego przysiku myszy pozwala na przybliżanie lub oddalanie obiektu poprzez przesuwanie kursora w pionie.

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

   void motion(GLsizei x, GLsizei y){
       delta_x = x - x_pos_old;
21     x_pos_old = x;

       delta_y = y - y_pos_old;
       y_pos_old = y;

       delta_zoom = y - zoom_pos_old;
       zoom_pos_old = y;

       glutPostRedisplay();
   }
31
   void draw(){
       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
       glLoadIdentity();

       if(status == 1){
```

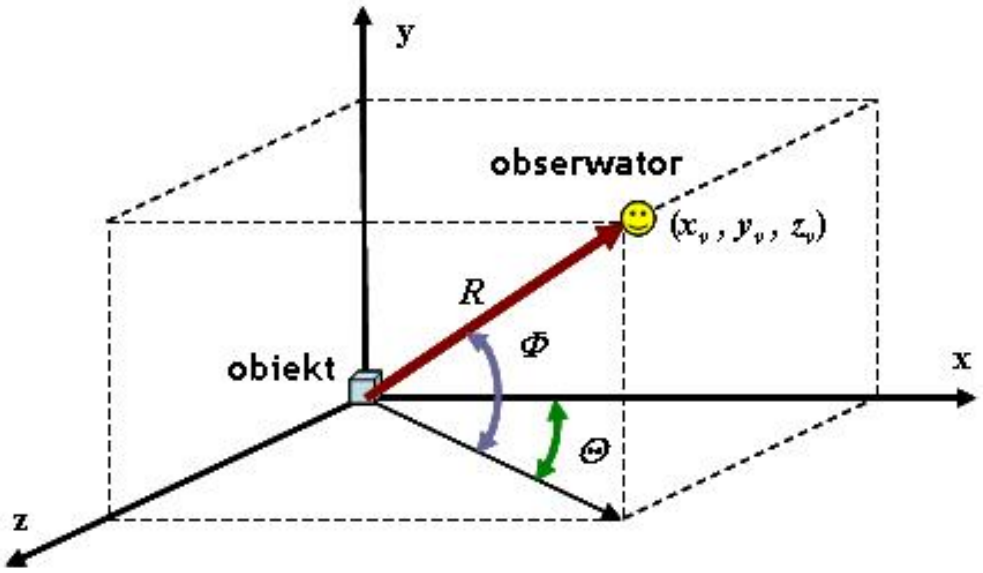
```
            theta[1] += delta_x*pix2angle;
            theta[0] += delta_y*pix2angle;
        } else if(status == 2){
            viewer[2] += delta_zoom/10.0;
            if(viewer[2] < 0) viewer[2] = 0;
        }

        gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)
            ;

        glRotatef(theta[0], 1.0, 0.0, 0.0);
        glRotatef(theta[1], 0.0, 1.0, 0.0);
        glRotatef(theta[2], 0.0, 0.0, 1.0);

        glColor3f(1.0, 1.0, 1.0);
        glutWireTeapot(3.0);

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



Rysunek 1: Czajnik obrócony wokół osi X i Y

3 Sterowanie położeniem obserwatora

W tym zadaniu zamiast czajnika wykorzystane zostało jajko z poprzedniego laboratorium. Program umożliwia zmiane położenia obserwatora za pomocą myszy. Przesuwanie kursora w pionie i w poziomie (wraz z wciśniętym lewym przyciskiem myszy) zmienia odpowiednie kąty określające pozycje obserwatora. Podobnie jak w zadaniu poprzednim przesuwanie kursora z wciśniętym prawym klawiszem myszy powoduje zoom obiektu.



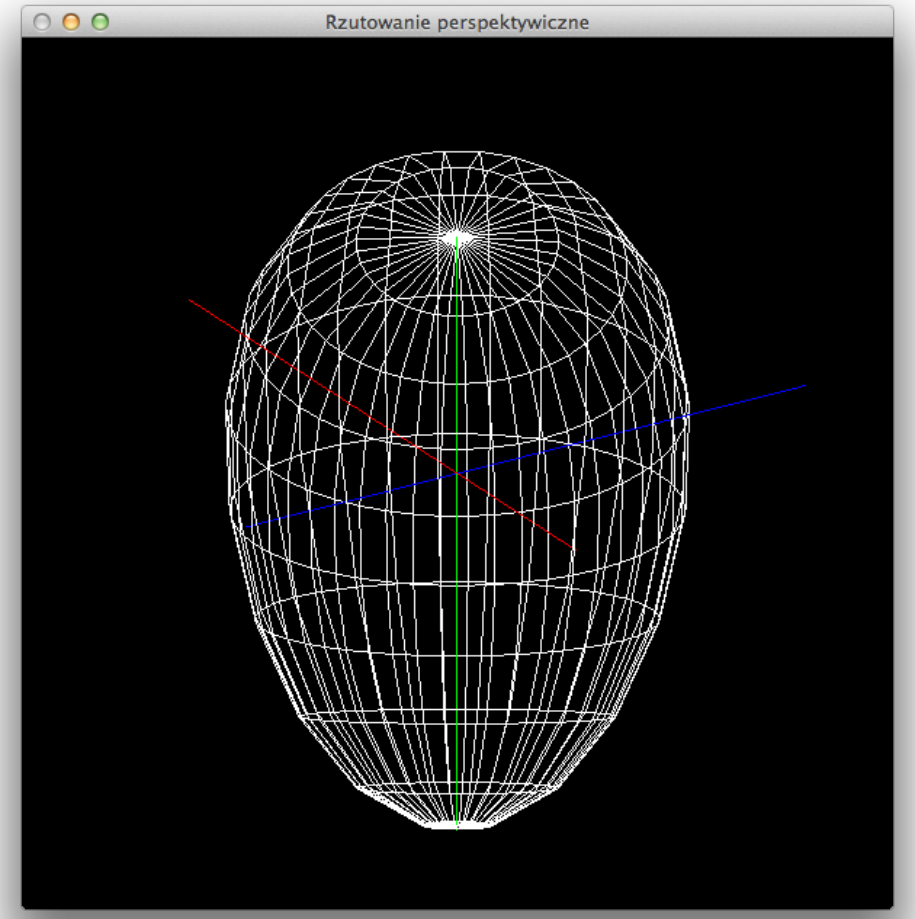
Rysunek 2: Położenie obserwatora względem obiektu

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

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_pos_old;
    zoom_pos_old = y;

    glutPostRedisplay();
}
```

```
    void draw(){
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
34      glLoadIdentity();

        if(status == 1){
            beta[0] += delta_x/40.0;
            beta[1] += delta_y/40.0;
        } else if(status == 2){
            r += delta_zoom/10.0;
        }

        if(beta[1] >= M_PI) beta[1] -= 2*M_PI;
44      else if(beta[1] <= -M_PI) beta[1] += 2*M_PI;

        if(beta[1] > M_PI/2 || beta[1] < -M_PI/2) yp = -1;
        else yp = 1;

        viewer[0] = r*cos(beta[0])*cos(beta[1]);
        viewer[1] = r*sin(beta[1]);
        viewer[2] = r*sin(beta[0])*cos(beta[1]);

        gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, yp, 0.0);
54
        glRotatef(theta[0], 1.0, 0.0, 0.0);
        glRotatef(theta[1], 0.0, 1.0, 0.0);
        glRotatef(theta[2], 0.0, 0.0, 1.0);

        glColor3f(1.0, 1.0, 1.0);
        egg();

        glFlush();
        glutSwapBuffers();
64  }
```

4 Wnioski

Wszystkie zadania zostały zrealizowane w czasie laboratorium. Największe trudności sprawiło przejście przez punkt krańcowy przy obrocie w zadaniu 2. Miało to związek z charakterystyką funkcji *sin*. Rozwiązaniem okazało się ograniczenie kąta Φ do wartości z przedziału $< -\pi; \pi >$, a następnie zmiana położenia obserwatora poprzez parametr funkcji *gluLookAt* w zależności od strony, na którą "patrzy" obserwator.



Rysunek 3: Zmiana położenia obserwatora