Sprawozdanie: Ćwiczenie nr. 4

Andrzej Borecki, indeks: 226205

28 stycznia 2019

1 Omówienie tematu

Na zajęciach zadaniem było zmienienie perspektywy rzutowanego obiektu (imbryczka). Wykonane to zostało przez obrót obiektu, a następnie alternatywnie przez obrót kamery, kiedy użytkownik poruszył myszą w danej osi jednoczenie przytrzymując wcinięty lewy przycisk myszy. Dodatkowo przytrzymanie wciśniętego prawego przycisku myszy i poruszanie nią powodowało odpowiednio przybliżenie bądź oddalenie obiektu w dopuszczalnym zakresie.

2 Omówienie kodu

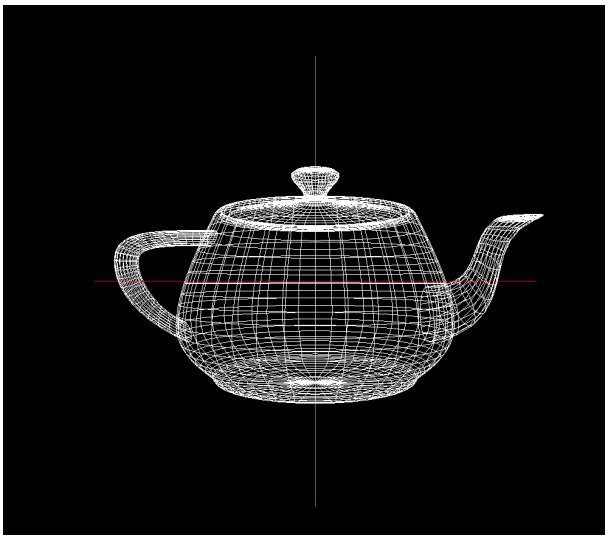
Poniższy kod realizuje wszystkie punkty z powyższego omówienia, nie udało mi się natomiast 'naprawić' błędu z obracaniem kamery.

```
static GLfloat theta = 0.0;    // kat obrotu obiektu
static GLfloat alfa = 0.0;      // kat obrotu obiektu
static GLfloat gamma = 1.0;      // poczatkowa skala
static GLfloat pix2angle;      // przelicznik pikseli na stopnie

static GLint status2 = 0;    // jak ponizej
static GLint status = 0;        // stan klawisza myszy LPM
                          // 0 - nie nacisnieto zadnego klawisza
                          // 1 - nacisniety zostal lewy klawisz

static int x_pos_old = 0;        // poprzednia pozycja kursora myszy
static int delta_x = 0;          // roznica pomiedzy pozycja biezaca
                          // i poprzednia kursora myszy w osi x
static int y_pos_old = 0;        // poprzednia pozycja kursora myszy
static int delta_y = 0;          // roznica pomiedzy pozycja biezaca
                          // i poprzednia kursora myszy w osi y

void Mouse(int btn, int state, int x, int y)
{
    if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        y_pos_old = y;          // przypisanie aktualnie odczytanej pozycji kursora
                          // jako pozycji poprzedniej
        status2 = 1;           // wcisniety zostal prawy klawisz myszy
    }
```

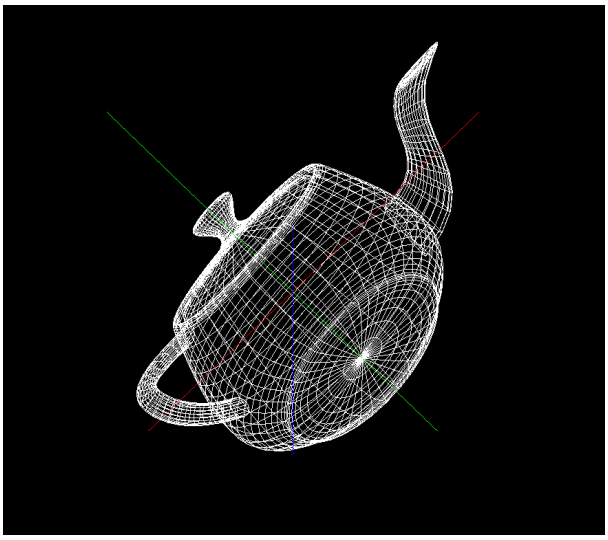
```
    else
        status2 = 0;           // nie zostal wcisniety klawisz RMB
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        x_pos_old = x;          // przypisanie aktualnie odczytanej pozycji kursora
                               // jako pozycji poprzedniej
        y_pos_old = y;          // przypisanie aktualnie odczytanej pozycji kursora
                               // jako pozycji poprzedniej
        status = 1;            // wcisniety zostal lewy klawisz myszy
    }
    else
        status = 0;            // nie zostal wcisniety klawisz LMB
}

void Motion(GLsizei x, GLsizei y)
{
    if (status == 1)
    {
        delta_x = x - x_pos_old;      // obliczenie roznicy polozenia kursora myszy
        x_pos_old = x;              // podstawienie biezacego polozenia jako poprzednie
        delta_y = y - y_pos_old;      // obliczenie roznicy polozenia kursora myszy
        y_pos_old = y;            // podstawienie biezacego polozenia jako poprzednie
    }

    if (status2 == 1)
    {
        delta_y = y - y_pos_old;      // obliczenie roznicy polozenia kursora myszy
        y_pos_old = y;            // podstawienie biezacego polozenia jako poprzednie
    }
    glutPostRedisplay();      // przerysowanie obrazu sceny
}

void RenderScene(void)
{

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Czyszczenie okna aktualnym kolorem czyszczacym

    glLoadIdentity();
    // Czyszczenie macierzy biezacej

    gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    // Zdefiniowanie polozenia obserwatora
    Axes();
    // Narysowanie osi przy pomocy funkcji zdefiniowanej powyzej

    if (status == 1)                     // jesli lewy klawisz myszy wcisniety
    {
        theta += delta_x * pix2angle;
        alfa += delta_y * pix2angle;     // modyfikacja kata obrotu o kat proporcjonalny
    }                                   // do roznicy polozen kursora myszy

    if (status2 == 1)                     // jesli prawy klawisz myszy wcisniety
    {
        gamma += delta_y * 0.008;    // spowolnienie zmiany
        if (R < 6.2)                 // wprowadzenie ograniczenia
                    R = 6.2;                        // najblizszej i najdalszej perspektywy
```

```
            if (R > 25)
                    R = 25;
    }

    viewer[0] = R * cos(theta*0.05) * cos(alfa*0.05);    // transformacje wspolrzednych kamery
    viewer[1] = R * sin(alfa*0.05);
    viewer[2] = R * sin(theta*0.05) * cos(alfa*0.05);

std::cout << " v0 " << viewer[0] << " v1 " << viewer[1] << " v2 "
    << viewer[2] << " R " << R << std::endl;
            // wypisanie wartosci kamery

    glRotatef(theta, 0.0, 1.0, 0.0);  //obrot obiektu o nowy kat
    glRotatef(alfa, 1.0, 0.0, 0.0);  //obrot obiektu o nowy kat
    glScaled(gamma, gamma, gamma); // skalowanie w kazdej osi *1.5

    glColor3f(1.0f, 1.0f, 1.0f);
    // Ustawienie koloru rysowania na bialy

    glutWireTeapot(3.0);
    // Narysowanie czajnika
    glFlush();
    // Przekazanie polecen rysujacych do wykonania
    glutSwapBuffers();

}
```

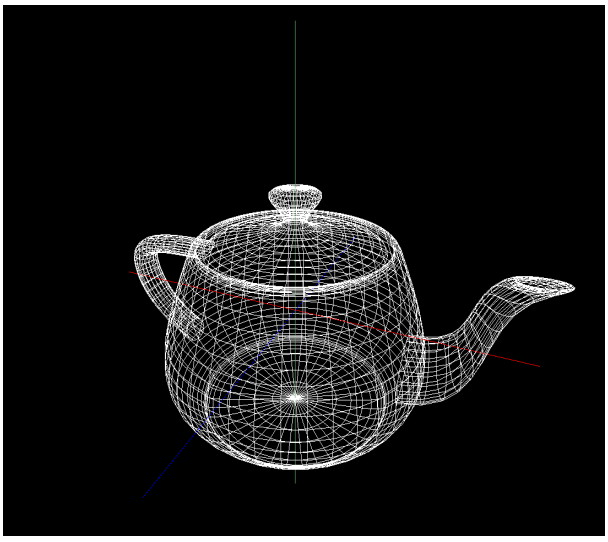
3 Rezultat prac



Rysunek 1: Pierwotny widok imbryka



Rysunek 2: Obrócony imbryczek



Rysunek 3: Zmiana perspektywy kamery z widokiem na imbryk