POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Grafika komputerowa i komunikacja człowiek - komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 7

TEMAT ĆWICZENIA

OpenGL – mini projekt (układ słoneczny)

Wykonał:	**Karol Pastewski 252798**
Termin:	**WT/TP 7.30-10.30**
Data wykonania ćwiczenia:	**18.01.2021r.**
Data oddania sprawozdania:	**25.01.2021r.**
Ocena:	

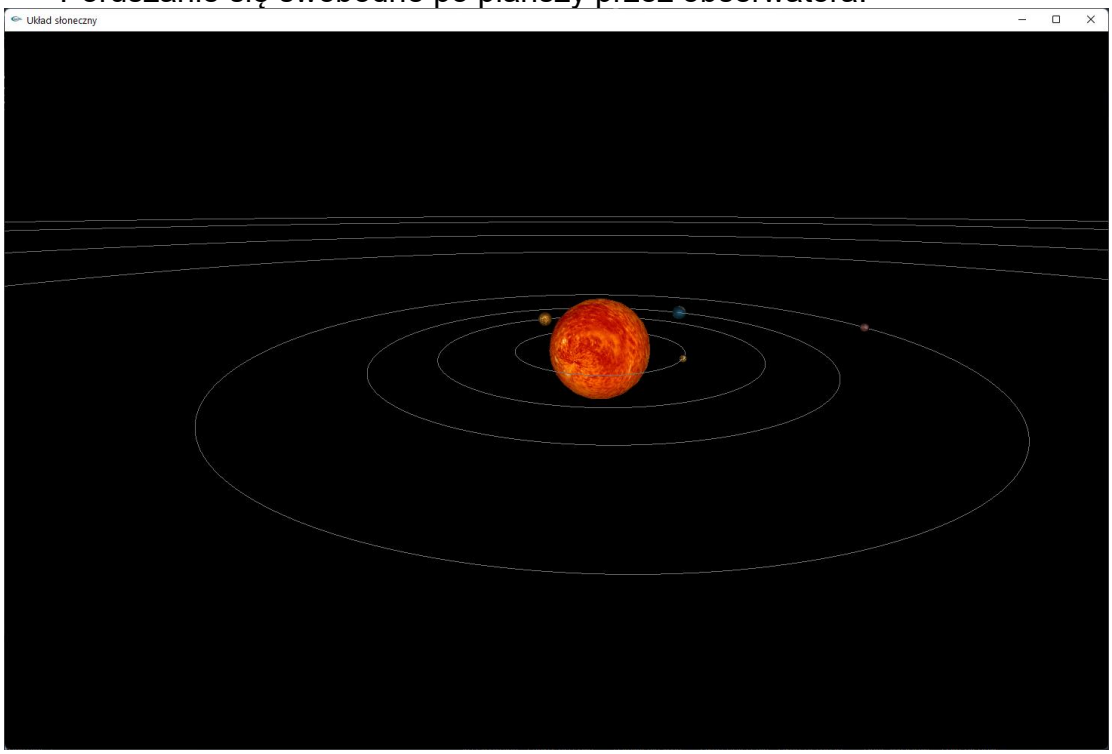
Uwagi prowadzącego:

1. Wstęp teoretyczny

1.1. Układ słoneczny

Zadaniem do wykonania było stworzenie mini projektu przy pomocy OpenGL, który będzie odzwierciedlał układ słoneczny. Funkcjonalności, które znalazły się w ostatecznej wersji projektu to:

- Narysowanie siedmiu planet i Słońca w formie kul.
- Narysowanie kulistych orbit wokół planet, które pokazują trajektorię poruszania się planet.
- Symulacja Słońca jako źródło światła.
- Teksturowanie Słońca oraz planet.
- Poruszanie się swobodne po planszy przez obserwatora.



Rysunek 1 Screenshot z działania programu

2. Nowe polecenia OpenGL

- **`GLUquadricObj* quadricObj = gluNewQuadric();`**
  Utworzenie nowej kwadryki (powierzchnia drugiego stopnia), która będzie potrzebna do narysowania kuli.
- **`gluQuadricDrawStyle(quadricObj, GLU_FILL);`**
  Określenie stylu rysowania dla kwadryki. GLU_FILL oznacza, że kwadryka będzie wyrenderowana z prymitywów.
- **`gluQuadricNormals(quadricObj, GLU_SMOOTH);`**
  Określenie jaki rodzaj wektora normlanego ma być stworzony dla kwadryki. GLU_SMOOTH oznacza, że dla każdego wierzchołka będzie utworzony wektor normalny.
- **`gluSphere(quadricObj, sun.getRadius(), 20, 20);`**
  Funkcja rysuje kule na ekranie. Przyjmuje parametr obiektu kwadryki, promień kuli (tutaj przykład dla Słońca). Ostatnie dwa parametry to „slices” (liczba podziałów wokół osi z, podobnych do linii długości geograficznej) oraz „stacks” (liczba podziałów wzdłuż osi z, podobnych do linii szerokości geograficznej).

3. Rozwiązanie zadania

```
62  Planet sun(radius[8], "Tekstury\\sun.tga");
63  Planet planets[] = {
64      Planet(radius[0], "Tekstury\\mercury.tga", distanceFromCenter[0], daysInYear[0]),
65      Planet(radius[1], "Tekstury\\venus.tga", distanceFromCenter[1], daysInYear[1]),
66      Planet(radius[2], "Tekstury\\earth.tga", distanceFromCenter[2], daysInYear[2]),
67      Planet(radius[3], "Tekstury\\mars.tga", distanceFromCenter[3], daysInYear[3]),
68      Planet(radius[4], "Tekstury\\jupiter.tga", distanceFromCenter[4], daysInYear[4]),
69      Planet(radius[5], "Tekstury\\saturn.tga", distanceFromCenter[5], daysInYear[5]),
70      Planet(radius[6], "Tekstury\\uranus.tga", distanceFromCenter[6], daysInYear[6]),
71      Planet(radius[7], "Tekstury\\neptune.tga", distanceFromCenter[7], daysInYear[7])
72  };
```

Na początku programu zdefiniowane są obiekty, które reprezentują planety. Oprócz Słońca (którego określa jego promień oraz tekstura), to do stworzenia obiektu potrzebny jest promień, tekstura, odległość od Słońca oraz liczba dni potrzebna do okrążenia Słońca. Utworzone obiekty planet są połączone w tabeli dla wygody przy późniejszym rysowaniu układy słonecznego.

```
136  sun.showTexture();
137  GLUquadricObj* quadricObj = gluNewQuadric();
138  gluQuadricDrawStyle(quadricObj, GLU_FILL);
139  gluQuadricNormals(quadricObj, GLU_SMOOTH);
140  gluQuadricTexture(quadricObj, GL_TRUE);
141  gluSphere(quadricObj, sun.getRadius(), 20, 20);
```

W powyższym kodzie widzimy przykład stworzenia kuli w układzie współrzędnych. Opis pojedynczych linii kodu jest w punkcie 2. W linii 136 jest wywoływana metoda obiektu Słońca, która wczytuje teksturę do buforu.

```
111  glDisable(GL_TEXTURE_2D);
112  for (double ring : rings) {
113      glColor3f(239, 195, 130);
114
115      glBegin(GL_LINE_LOOP);
116      for (int i = 0; i < 3600; i++) {
117          float angle = 1.0 * float(i) / float(3600);
118          float x = ring * cos(2 * M_PI * angle);
119          float z = ring * sin(2 * M_PI * angle);
120
121          glVertex3f(x, 0, z);
122      }
123      glEnd();
124  }
125  glEnable(GL_TEXTURE_2D);
```

Zamieszczony powyżej fragment kodu rysuje pierścienie wokół Saturna. Promienie tych pierścieni są określone na początku programu i są przechowywane w tablicy rings. Należy pamiętać, aby przed rysowaniem wyłączyć teksturowanie.

```
91  void drawOrbit(Planet planet) {
92      glDisable(GL_TEXTURE_2D);
93      glColor3f(0.6, 0.6, 0.6);
94      glBegin(GL_LINE_LOOP);
95      for (int i = 0; i < 360; i++) {
96          float angle = 1.0 * float(i) / float(360);
97          float x = planet.getDistanceFromCenter() * cos(2 * M_PI * angle);
98          float y = planet.getDistanceFromCenter() * sin(2 * M_PI * angle);
99
100         glVertex3f(x, 0, y);
101     }
102     glEnd();
103     glEnable(GL_TEXTURE_2D);
104     glColor3f(1.0, 1.0, 1.0);
105 }
```

Ta funkcja rysuje dla podanej w parametrze planety jej orbitę. Funkcja ta wywoływana jest dla każdej planety i w wykonaniu jest podobna do rysowania pierścieni. Także tutaj trzeba pamiętać o wyłączaniu teksturowania.

```
201  void changeDay() {
202      day += 0.1;
203      Sleep(1);
204      renderScene();
205
206      if (day > 60223) {
207          day -= 60223;
208      }
209  }
```

```
437  glutIdleFunc(changeDay);
```

Funkcja changeDay() wykonuje się zawsze gdy program jest w stanie spoczynku i powoduje zmianę dnia o 0.1 (większa wartość oznaczałaby zwiększenie poruszania się planet wokół Słońca).