



Politechnika Wrocławska

Sprawozdanie 6

Ćwiczenie 6.Układ słoneczny

Krzysztof Zalewa

7.1.2025

Spis treści

1 Wstęp teoretyczny	2
1.1 Układ słoneczny	2
1.2 Wykorzystane tekstyury	2
2 Zadanie laboratoryjne	8
2.1 Treść zadania	8
2.2 Opis działania programu	8
2.3 Kod programu	8
3 Wnioski	17
4 Źródła	17

1 Wstęp teoretyczny

1.1 Układ słoneczny

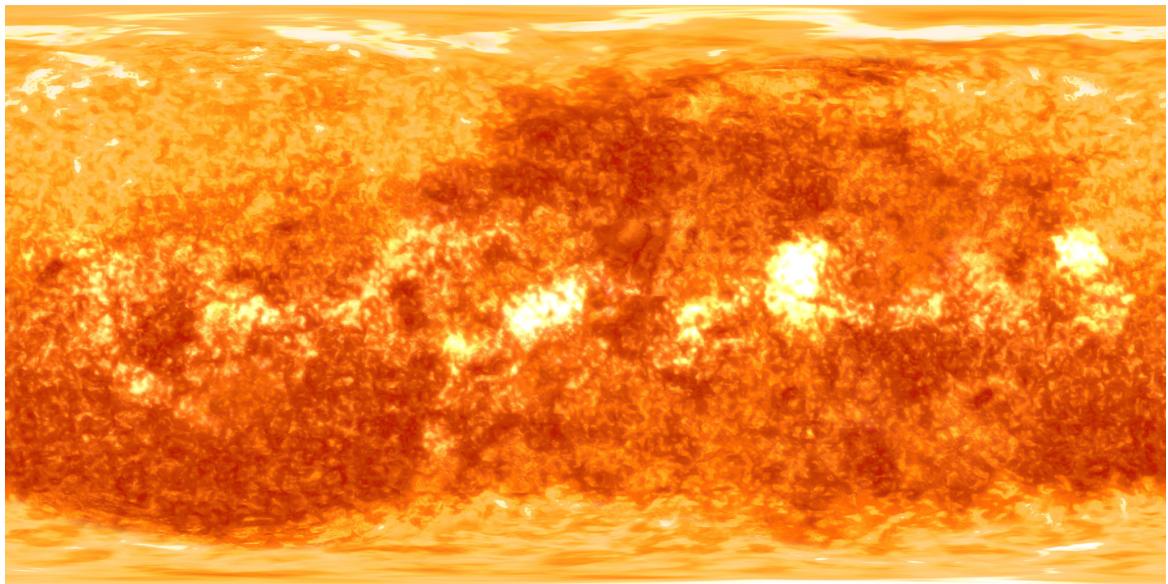
Wielkości planet zostały dobrane eksperymentalnie "tak żeby wyglądały dobrze". Wyjątkiem są wartości pochylenia osiowego planet (dokładność do 1 miejsca po przecinku). Wysoki stopień

Planeta	Moja wartość	Wartość rzeczywista
Merkury	0.03	0.034
Wenus	177.4*	177.36
Ziemia	23.5	23.44
Mars	25.2	25.19
Jowisz	3.1	3.13
Saturn	26.7	26.73
Uran	97.8	97.77
Neptun	28.3	28.32
Pluton	122.5*	122.53

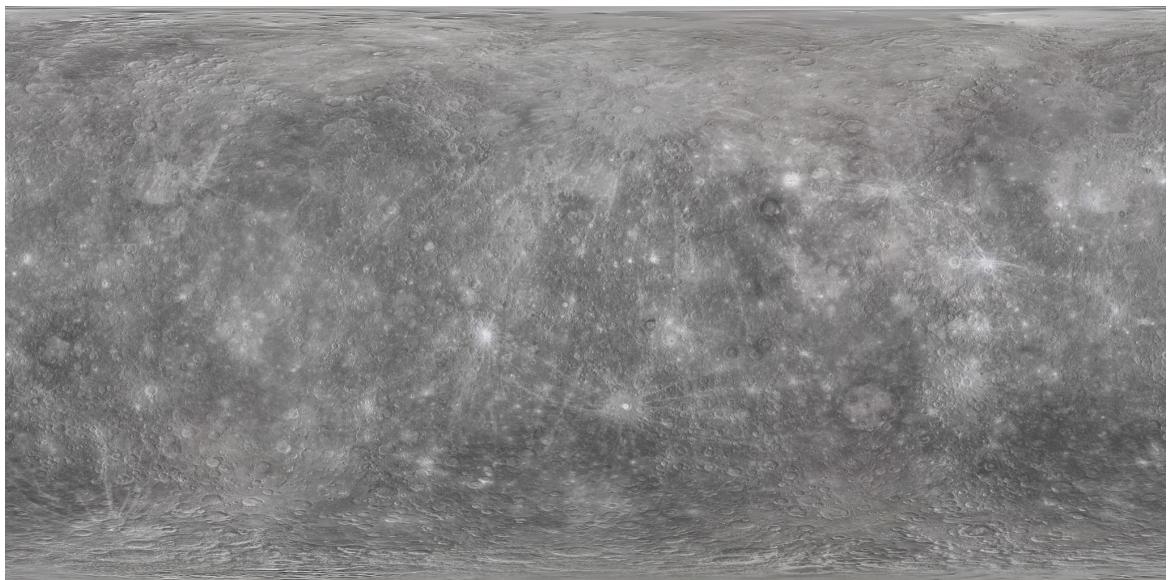
Table 1: Pochylenie osiowe planet

nachylenia sprawia że Wenus i Pluton obracają się zgodnie z ruchem wskazówek zegara. Pozostałe planety obracają się w przeciwną stronę.

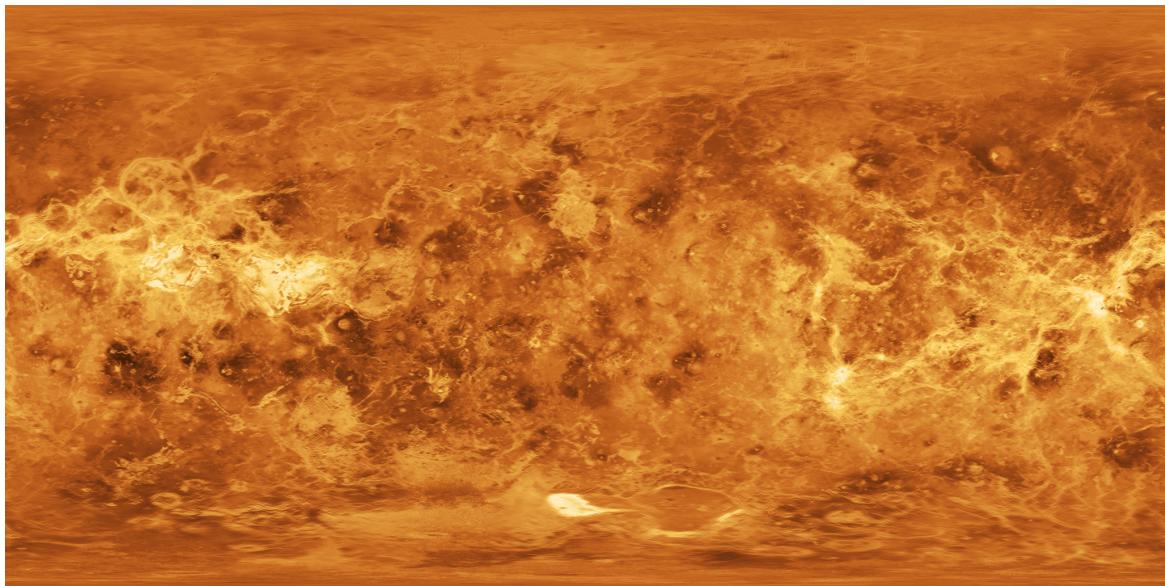
1.2 Wykorzystane tekstyury



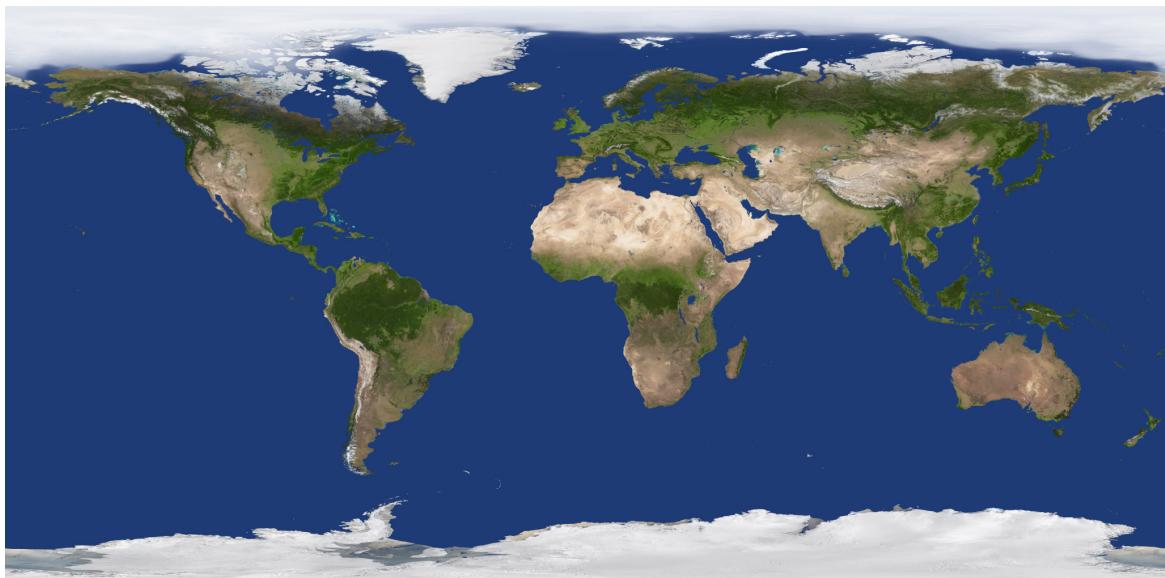
Rysunek 1: Tekstura słońca[1]



Rysunek 2: Tekstura merkurego[1]



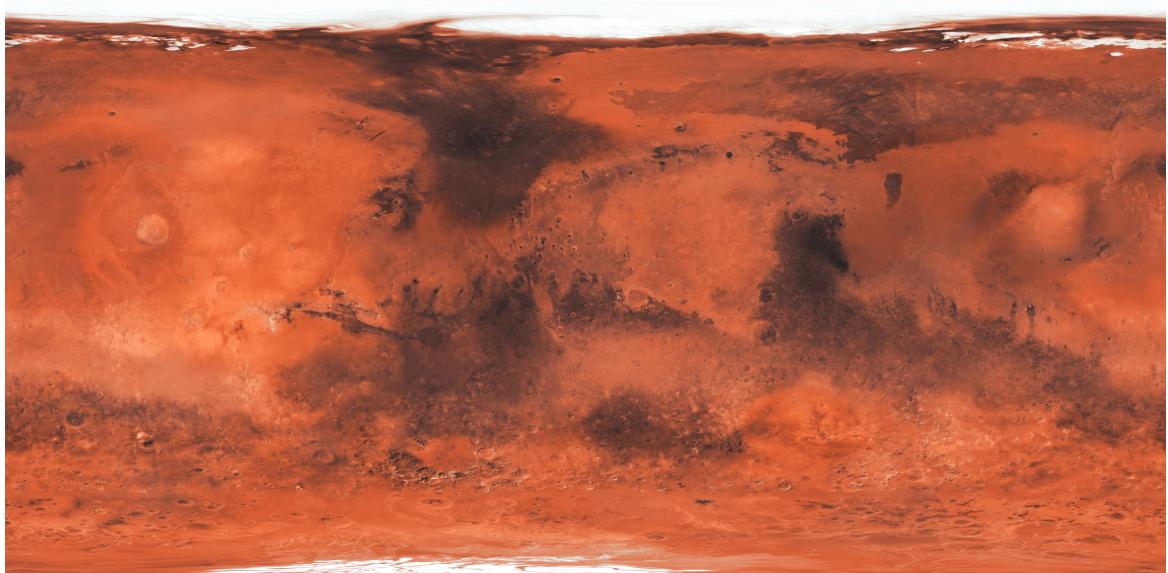
Rysunek 3: Tekstura wenus[1]



Rysunek 4: Tekstura ziemi[1]



Rysunek 5: Tekstura księżyca[1]



Rysunek 6: Tekstura marsa[1]



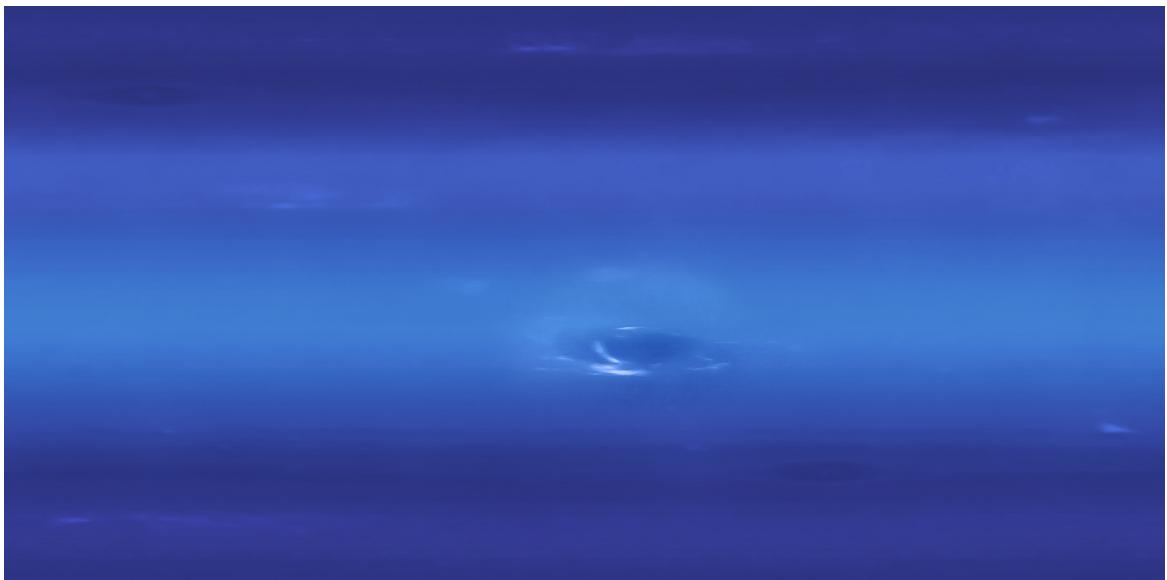
Rysunek 7: Tekstura jowisza[1]



Rysunek 8: Tekstura saturna[2]



Rysunek 9: Tekstura urana[1]



Rysunek 10: Tekstura neptuna[1]



Rysunek 11: Tekstura pluta[2]

2 Zadanie laboratoryjne

2.1 Treść zadania

W ramach zadania laboratoryjnego należało napisać program symulujący układ słońeczny. Symulacja miała składać się z planet wirujących wokół własnej osi Y i jednocześnie orbitujących wokół słońca. Orbita powinna być eliptyczna ze słońcem w jednym z ognisk. Ruch po orbicie powinien uwzględniać 3 prawo Keplera. Słońce powinno być źródłem światła.

2.2 Opis działania programu

Zgodnie z treścią zadania program rysuje 12 obiektów (10 planet, słońce i Księżyc). Wszystkie planety obracają się wokół własnej osi i orbitują wokół słońca. **F1** - Zmiana kamery

F2 - Poprzednia planeta

F3 - Następna planeta

F4 - Wypisanie listy planet

ESC - Wyjście z programu

Ruch myszy w osi X - Obrót kamery w osi X

Ruch myszy w osi Y - Obrót kamery w osi Y

Scroll up - Przybliżenie obiektu

Scroll down - Oddalenie obiektu

2.3 Kod programu

```
1 #include <windows.h>
2 #include <iostream>
3 #include <GL/glut.h>
4 #include <vector>
5 #define FREEGLUT_STATIC
6 #include <GL/freeglut.h>
7 #define STB_IMAGE_IMPLEMENTATION
8 #include "stb_image.h"
```

```

9 //My .h files
10 #include "Planet.h"
11 #include "Sun.h"
12
13 using namespace std;
14 //Constants
15 const int TEXT_HEIGHT = 13;
16 const int PLANET_NUM = 9;
17 const int ALL = 11;
18 const int CAMERA_NUM = 2;
19 //Global variables
20 HWND consoleWindow;
21 HWND glutWindow;
22 GLuint textureIDs[ALL];
23 void *font = GLUT_BITMAP_8_BY_13;
24 int currentPlanet = 3;
25 Planet planets[PLANET_NUM];
26 string planetNames[PLANET_NUM+1] = {"Mercury","Venus","Earth","Moon","Mars","Jupiter",
27                                     "Saturn","Uranus","Neptune","Pluto"};
27 const char* fileNames[ALL] = {"textures\\2k_sun.tga","textures\\2k_mercury.tga","textures\\2k_venus_surface.tga",
28                               "textures\\2k_earth_daymap.tga",
29                               "textures\\2k_moon.tga","textures\\2k_mars.tga","textures\\2k_jupiter.tga",
29                               "textures\\saturnmapthumb.tga","textures\\2k_uranus.tga",
29                               "textures\\2k_neptune.tga","textures\\plutomapthumb.tga"};
30
31 float planetSizes[PLANET_NUM] = {2,6.5,7,6.5,14,10,8,8,5};
32 float planetDistances[PLANET_NUM] =
33     {15.0,25.0,35.5,50.0f,70.0f,80.0f,90.0f,100.0f,110.0f};
33 float planetAxialTilts[PLANET_NUM] =
34     {0.03f,177.4f,23.5f,25.2f,3.1f,26.7f,97.8f,28.3f,122.5f};
34 float xAxis[PLANET_NUM] = {1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f};
35 float yAxis[PLANET_NUM] = {1.3f, 1.3f, 1.2f, 1.21f, 1.25f, 1.15f, 1.25f, 1.2f,
35     1.4f};
36 float planetOrbitalTilts[PLANET_NUM] =
37     {7.0,3.39,0.0,1.85,1.31,2.49,0.77,1.77,17.16} ;
37 int currentCamera = 0;
38 string cameraNames[CAMERA_NUM] = {"Wolna","Na planecie"};
39 float pix2angle = 360.0/800,theta = 0.0f,phi = 0.0f;
40 int radius = 35,lastX = 0,lastY = 0;
41 float cameraRotationX = radius * cosf((theta*(M_PI/180))) * cosf((phi*(M_PI/180)));
42 float cameraRotationY = radius * sinf((phi*(M_PI/180)));
43 float cameraRotationZ = radius * sinf((theta*(M_PI/180))) * cosf((phi*(M_PI/180)));
44 float cameraX = 0,cameraY = 0;
45 Sun sun;
46 //Window controls
47 void toggleFocusToConsole() {
48     ShowWindow(glutWindow, SW_HIDE);
49     ShowWindow(consoleWindow, SW_SHOWNORMAL);
50     SetForegroundWindow(consoleWindow);
51 }
52 void toggleFocusToGLUT() {
53     ShowWindow(consoleWindow, SW_HIDE);
54     ShowWindow(glutWindow, SW_SHOWNORMAL);
55     SetForegroundWindow(glutWindow);
56 }
57 //UI

```

```

58 void drawString(const char *str, int x, int y, float color[4], void *font){
59     glPushAttrib(GL_LIGHTING_BIT | GL_CURRENT_BIT);
60     glDisable(GL_LIGHTING);
61     glDisable(GL_TEXTURE_2D);
62     glDepthFunc(GL_ALWAYS);
63     glColor4fv(color);
64     glRasterPos2i(x, y);
65     while(*str){
66         glutBitmapCharacter(font, *str);
67         ++str;
68     }
69     glEnable(GL_TEXTURE_2D);
70     glEnable(GL_LIGHTING);
71     glDepthFunc(GL_EQUAL);
72     glPopAttrib();
73 }
74 void showInfo(){
75     glPushMatrix();
76     glLoadIdentity();
77     glMatrixMode(GL_PROJECTION);
78     glPushMatrix();
79     glLoadIdentity();
80     gluOrtho2D(0, 800, 0, 800);
81     float color[4] = {1, 1, 1, 1};
82     string s = "Kamera: "+cameraNames[currentCamera];
83     drawString(s.c_str(),2,800-TEXT_HEIGHT,color,font);
84     s = "F1 - Zmien kamere";
85     drawString(s.c_str(),2,800-(2*TEXT_HEIGHT),color,font);
86     s = "Planeta: "+planetNames[currentPlanet];
87     drawString(s.c_str(),2,800-(3*TEXT_HEIGHT),color,font);
88     s = "F2 - Poprzednia planeta || F3 Nastepna planeta || F4 - Lista Planet";
89     drawString(s.c_str(),2,800-(4*TEXT_HEIGHT),color,font);
90     s = "ESC - Wyjdz z programu";
91     drawString(s.c_str(),2,800-(5*TEXT_HEIGHT),color,font);
92     glPopMatrix();
93     glMatrixMode(GL_MODELVIEW);
94     glPopMatrix();
95 }
96 //Keyboard control
97 void normalKey(u_char key,int x,int y){
98     float moveSpeed = 10.0f;
99     switch (key){
100     case 'W':
101     case 'w':
102         if(currentCamera == 0){
103             cameraY += 10.0f;
104         }
105         break;
106     case 'A':
107     case 'a':
108         if(currentCamera == 0){
109             cameraX += 10.0f;
110         }
111         break;
112     case 'S':
113     case 's':

```

```

114         if(currentCamera == 0){
115             cameraY -= 10.0f;
116         }
117         break;
118     case 'D':
119     case 'd':
120         if(currentCamera == 0){
121             cameraX -= 10.0f;
122         }
123         break;
124     case 27:
125         exit(0);
126         break;
127     default:
128         break;
129     }
130 }
131 void specialKey(int key,int x,int y){
132     switch (key){
133         //F1 - Change camera
134     case GLUT_KEY_F1:
135         if(currentCamera==0){
136             currentCamera = 1;
137         }else{
138             currentCamera = 0;
139         }
140         break;
141         //F2 - Prevoius planet
142     case GLUT_KEY_F2:
143         planets[currentPlanet].setCamera();
144         currentPlanet--;
145         planets[currentPlanet].setCamera();
146         if(currentPlanet == 0){
147             currentPlanet = PLANET_NUM-1;
148         }
149         break;
150         //F3 - Next planet
151     case GLUT_KEY_F3:
152         planets[currentPlanet].setCamera();
153         currentPlanet++;
154         planets[currentPlanet].setCamera();
155         if(currentPlanet == PLANET_NUM){
156             currentPlanet = 1;
157         }
158         break;
159         //F4 - Print plantets
160     case GLUT_KEY_F4:
161         for(int i=0;i<PLANET_NUM;i++){
162             cout<<i<<" . "<<planetNames[i]<<"\n";
163         }
164         toggleFocusToConsole();
165         cout<<"Nacisnij Enter zeby kontynuowac\n"<<flush;
166         cin.get();
167         cin.get();
168         toggleFocusToGLUT();
169         break;

```

```

170     default:
171         break;
172     }
173     glutPostRedisplay();
174 }
175 void mouse(int x, int y){
176     float dY = y - lastY;
177     lastY = y;
178     float dX = x - lastX;
179     lastX = x;
180     theta += dX * pix2angle;
181     phi += dY * pix2angle;
182     if (phi > 89.0f) {phi = 89.0f;}
183     if (phi < -89.0f) {phi = -89.0f;}
184     switch(currentCamera){
185         case 0:
186             cameraRotationX = radius * cosf((theta*(M_PI/180.0f))) *
187             ↵   cosf((phi*(M_PI/180.0f)));
188             cameraRotationY = radius * sinf((phi*(M_PI/180.0f)));
189             cameraRotationZ = radius * sinf((theta*(M_PI/180.0f))) *
190             ↵   cosf((phi*(M_PI/180.0f)));
191             break;
192         case 1:
193             break;
194     }
195     lastX = x;
196     lastY = y;
197     glutPostRedisplay();
198 }
199 void mouseWheel(int button, int dir, int x, int y){
200     if (dir > 0){
201         radius -= 10;
202     }else{
203         radius += 10;
204     }
205     if(radius<20){
206         radius=20;
207     }
208     glutPostRedisplay();
209 }
210 void timer(int value) {
211     glutPostRedisplay();
212     glutTimerFunc(16, timer, 0);
213 }
214 void animate(){
215     for(int i = 0;i<PLANET_NUM;i++){
216         planets[i].animateSpin();
217         planets[i].move(0.1f);
218     }
219     planets[2].animateSpinMoon();
220     planets[2].moveMoon(0.1f);
221     glutPostRedisplay();
222 }
223 void display(){
    GLfloat lPos[] = {0,0,0,1};

```

```

224     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
225     glLoadIdentity();
226     glTranslatef(-cameraX,0.0f,-cameraY);
227     if(currentCamera==0){
228         gluLookAt(cameraRotationX,cameraRotationY,cameraRotationZ,0,0,0,0,1,0);
229     }else{
230         planets[currentPlanet].setCamera();
231     }
232     glLightfv(GL_LIGHT0,GL_POSITION,lPos);
233     sun.draw(textureIDs[0]);
234     for(int i = 0;i<PLANET_NUM;i++){
235         glPushMatrix();
236         glRotatef(planetOrbitalTilts[i], 1.0f, 0.0f, 0.0f);
237         planets[i].drawOrbit();
238         planets[i].draw(textureIDs[i+1]);
239         glPopMatrix();
240     }
241     showInfo();
242     glutSwapBuffers();
243 }
244 void loadTexture(const char* fileName,GLuint texID){
245     glGenTextures(1,&textureIDs[texID]);
246     glBindTexture(GL_TEXTURE_2D, textureIDs[texID]);
247     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
248     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
249     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
250     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
251     int width,height,nrChannels;
252     unsigned char *data = stbi_load(fileName, &width, &height, &nrChannels, 0);
253     if (data){
254         glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
255             GL_UNSIGNED_BYTE, data);
256         cout << "Texture loaded successfully: " << fileName << endl;
257     }else{
258         cout << "Failed to load texture! "<< fileName << " "
259         <<stbi_failure_reason()<< endl;
260         system("pause");
261         exit(1);
262     }
263     stbi_image_free(data);
264 }
265 void init(){
266     sun = Sun(GL_LIGHT0,0,16);
267     float pos = 10;
268     for(int i = 0;i<PLANET_NUM;i++){
269         if(i!=3){
270             planets[i] = Planet(i,planetSizes[i]);
271             pos += planetDistances[i];
272             planets[i].setDistance(pos);
273             planets[i].setEllipse(xAxis[i],yAxis[i]);
274             planets[i].setTilt(planetAxialTilts[i]);
275         }
276     }
277     moon newMoon = {
278         25,2,0,13,planets[2].getPosition(),6.7
279     };

```

```

278     planets[2].addMoon(newMoon);
279     glEnable(GL_DEPTH_TEST); //bez tego frontalna sciana nadpisuje tylnią
280     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
281     glMatrixMode(GL_PROJECTION);
282     gluPerspective(45,1,0.01,5000);
283     glMatrixMode(GL_MODELVIEW);
284     glShadeModel(GL_SMOOTH);
285     glEnable(GL_LIGHTING); //Włączenie oświetlenia
286     glEnable(GL_LIGHT0); //Dodanie źródła światła
287     glEnable(GL_TEXTURE_2D); //Włącza teksturowanie
288     glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
289     for(int i = 0;i<ALL;i++){
290         loadTexture(fileNames[i],i);
291     }
292     planets[2].setMoonTexture(textureIDs[4]);
293 }
294 //Na 7 stycznia
295 //TODO - 2 Kamery sterowane przy użyciu myszy
296 //TODO - Swobodna na sferze (biegunowa,azymut)
297 //TODO - Umiejscowiona na planecie
298 int main(int argc, char** argv){
299     consoleWindow = GetConsoleWindow();
300     glutInit(&argc, argv);
301     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
302     glutInitWindowSize(800,800);
303     glutCreateWindow("Lab 6 - Uklad sloneczny");
304     glutWindow = FindWindowW(NULL,L"Lab 6 - Uklad sloneczny");
305     init();
306     glutDisplayFunc(display);
307     glutKeyboardFunc(normalKey);
308     glutSpecialFunc(specialKey);
309     glutMotionFunc(mouse);
310     glutMouseWheelFunc(mouseWheel);
311     glutIdleFunc/animate);
312     glutTimerFunc(25, timer, 0);
313     glutMainLoop();
314     return 0;
315 }
```

Fragment kodu 1: Fragment kodu z programu

```

1 #include "Planet.h"
2 #include <math.h>
3 #define FREEGLUT_STATIC
4 #include <GL/freeglut.h>
5 void Planet::axis(){
6     glDisable(GL_LIGHTING);
7     glBegin(GL_LINES);
8     glColor3f(1.0, 0.0, 0.0);
9     glVertex3f(-15.0, 0.0, 0.0);
10    glVertex3f(15.0, 0.0, 0.0);
11    glColor3f(0.0, 1.0, 0.0);
12    glVertex3f(0.0, -15.0, 0.0);
13    glVertex3f(0.0, 15.0, 0.0);
```

```

14     glColor3f(0.0, 0.0, 1.0);
15     glVertex3f(0.0, 0.0, -15.0);
16     glVertex3f(0.0, 0.0, 15.0);
17     glEnd();
18     glEnable(GL_LIGHTING);
19 }
20 void Planet::animateSpin(){
21     spin += planetSpeed;
22     if (spin > 360.0f) {
23         spin -= 360.0f;
24     }
25 }
26 void Planet::move(float dTime){
27     float orbitalPeriod = sqrtf(powf(distance, 3));
28     float orbitalSpeed = 2.0f * 3.1415926f / orbitalPeriod;
29     planetTheta += orbitalSpeed*dTime;
30     if (planetTheta >= 2.0f * 3.1415926f) {
31         planetTheta -= 2.0f * 3.1415926f;
32     }
33 }
34 void Planet::moveMoon(float dTime){
35     moonTheta += moonOrbitSpeed*dTime;
36     if (spin > 360.0f) {
37         spin -= 360.0f;
38     }
39 }
40 void Planet::draw(u_int texture){
41     glPushMatrix();
42     glColor3f(1.0, 1.0, 1.0);
43     float planetX = xAxis * distance * cosf(planetTheta) + ellipseCenterX;
44     float planetY = yAxis * distance * sinf(planetTheta) - ellipseCenterY;
45     glTranslatef(planetX, 0.0f, planetY);
46     if(hasMoon){
47         float x = myMoon.distance * cosf(moonTheta);
48         float y = myMoon.distance * sinf(moonTheta);
49         glPushMatrix();
50         drawMoonsOrbit();
51         glRotatef(moonTheta, 0.0f, 1.0f, 0.0f);
52         glTranslatef(x, 0.0f, y);
53         glRotatef(tilt, 1.0f, 0.0f, 0.0f);
54         glRotatef(myMoon.spin, 0.0f, 1.0f, 0.0f);
55         glEnable(GL_TEXTURE_2D);
56         glBindTexture(GL_TEXTURE_2D, moonTexture);
57         GLUquadric *quad = gluNewQuadric();
58         gluQuadricTexture(quad, GL_TRUE);
59         gluSphere(quad, myMoon.size, myMoon.points, myMoon.points);
60         gluDeleteQuadric(quad);
61         glDisable(GL_TEXTURE_2D);
62         glPopMatrix();
63     }
64     positon = {planetX, 0.0f, planetY};
65     glRotatef(90.0, 1.0f, 0.0f, 0.0f);
66     glRotatef(tilt, 1.0f, 0.0f, 0.0f);
67     glRotatef(spin, 0.0f, 0.0f, 1.0f);
68     // axis();
69     if(camera){

```

```

70         gluLookAt(planetX, 0.0f, planetY,0,0,0,0,1,0);
71     }
72     glEnable(GL_TEXTURE_2D);
73     glBindTexture(GL_TEXTURE_2D, texture);
74     GLUquadric *quad = gluNewQuadric();
75     gluQuadricTexture(quad, GL_TRUE);
76     gluSphere(quad,size,points,points);
77     gluDeleteQuadric(quad);
78     glDisable(GL_TEXTURE_2D);
79     glPopMatrix();
80 }
81 void Planet::drawOrbit(){
82     glPushMatrix();
83     glDisable(GL_LIGHTING);
84     glPointSize(2.0f);
85     glBegin(GL_POINTS);
86     glColor3f(1.0, 1.0, 1.0);
87     for(int i = 0;i<points *2;i++){
88         float theta = 2.0f * 3.1415926f * float(i) / float(points * 2);
89         float x = xAxis * distance * cosf(theta);
90         float y = yAxis * distance * sinf(theta);
91         glVertex3f(x + ellipseCenterX, 0.0f, y - ellipseCenterY);
92     }
93     glEnd();
94     glEnable(GL_LIGHTING);
95     glPopMatrix();
96 }
97 void Planet::drawMoonsOrbit(){
98     glPushMatrix();
99     glDisable(GL_LIGHTING);
100    glPointSize(2.0f);
101    glBegin(GL_POINTS);
102    glColor3f(1.0, 1.0, 1.0);
103    for(int i = 0;i<myMoon.points *2;i++){
104        float theta = 2.0f * 3.1415926f * float(i) / float(myMoon.points * 2);
105        float x = myMoon.distance * cosf(theta) ;
106        float y = myMoon.distance * sinf(theta) ;
107        glVertex3f(x,0.0f,y);
108    }
109    glEnd();
110    glEnable(GL_LIGHTING);
111    glPopMatrix();
112 }
113 void Planet::animateSpinMoon(){
114     myMoon.spin += moonSpeed;
115     if (myMoon.spin > 360.0f) {
116         myMoon.spin -= 360.0f;
117     }
118 }
119 void Planet::setTilt(float degree){
120     tilt = degree;
121 }
122 Planet::Planet(float newDistance,float newSize){
123     distance = newDistance;
124     size = newSize;
125 }
```

```

126 Planet::Planet() : points(50), size(1.0f), distance(0.0f) {}
127 void Planet::setDistance(float newDistance){
128     distance = newDistance;
129     positon = {distance, 0.0f, 0.0f};
130 }
131 void Planet::setEllipse(float newX,float newY){
132     xAxis = newX;
133     yAxis = newY;
134     c = sqrtf(yAxis * yAxis - xAxis * xAxis);
135     ellipseCenterX = 0.0f;
136     ellipseCenterY = c* distance/3;
137 }
138 point Planet::getPosition(){
139     return positon;
140 }
141 void Planet::addMoon(moon newMoon){
142     myMoon = newMoon;
143     hasMoon = true;
144 }
145 void Planet::setCamera(){
146     camera = !camera;
147 }
148 void Planet::setMoonTexture(u_int newTexture){
149     moonTexture = newTexture;
150 }
```

Fragment kodu 2: Fragment kodu z programu

```

1 #include "Sun.h"
2 void Sun::initLight(){
3
4 }
5 void Sun::draw(u_int texture){
6     glDisable(GL_LIGHTING);
7     Planet::draw(texture);
8     glEnable(GL_LIGHTING);
9 }
```

Fragment kodu 3: Fragment kodu z programu

3 Wnioski

4 Źródła

1. <https://www.solarsystemscope.com/textures/>
2. <https://planetpixelemporium.com/pluto.html>