**Prelucrare grafica**

**Documentatie proiect**

Giesswein Alexia

Grupa: 30231

Prof: Constantin Nandra

**Cuprins**

1. **Prezentarea temei**……………………………………….3
2. **Scenariul**…………………………………………………3
   1. **Descrierea scenei si a obiectelor**…………………...3
   2. **Functionalitati**………………………………………3
3. **Detalii de implementare**………………………………...4
   1. **Functii si algoritmi**…………………………………4
   2. **Modelul grafic**………………………………………7
   3. **Ierarhia de clase**…………………………………….8
4. **Prezentarea interfetei grafice/manualul de utilizare**….8
5. **Concluzii**………………………………………..………12
6. **Bibliografie**……………………………………………..12
7. **Prezentarea temei**

Scopul acestui proiect este de a reprezenta o scena grafica care sa fie cat mai realista, complet functionala, care contine obiecte 3D incarcate din blender, utilizand OpenGL pentru diverse efecte, cum ar fi lumini, umbre, animatii. Folosind tastatura si mouse-ul, utilizatorul se poate plimba prin intreaga scena. Scena din proiect reprezinta un peisaj din natura, o insula.

1. **Scenariul**
   1. **Descrierea scenei si a obiectelor**

Scena din proiect reprezinta o insula, cu munti, copaci, pietre si care e inconjurata de apa. Alte obiecte adaugate sunt: o cupola in care se afla o torta care lumineaza, un felinar care lumineaza, un catel care se poate roti (se roteste in jurul cozii), un templu, un avion care zboara (se misca in linie dreapta), o lanterna care lumineaza si niste flori care ilustreaza efectul de transparenta. De asemenea, avem un cub care este lumina directionala care lumineaza intreaga scena, iar luminile punctiforme sunt la felinar, lanterna si torta.

* 1. **Functionalitati**

Sunt mai multe functionalitati implementate in proiect. Miscarea prin scena se realizeaza cu ajutorul tastaturii, a tastelor “WASD” si cu ajutorul mouse-ului. Sunt 4 surse de lumina, una directionala care ilustreaza si umbra (miscand cubul- sursa de lumina directionala, se misca si umbrele) si 3 surse de lumina punctiforme. Lumina directionala este plasata deasupra scenei, iar luminile punctiforme sunt plaste la torta, felinar si lanterna, acestea din urma pot fi pornite si oprite la apasarea unui buton. De asemenea, in scena se regasesc si doua animatii, un avion care zboara inainte si un catel care se roteste in jurul cozii. Avionul zboara incontinuu si poate fi readus la pozitia initiala apasand o tasta, iar caine se invarte doar cand se apasa tasta. Sunt implementate si 3 efecte: ceata care se activeaza si dezactiveaza de la apasarea unui buton, transparenta care este ilustrata la una dintre flori si la felinar si cele 3 surse de lumina punctiforma. Mai este implementat si modul de vizualizare al scenei, care merge pe o directie. Scena poate fi vizualizata in modurile solid, wireframe si pointframe.

1. **Detalii de implementare**
   1. **Functii si algoritmi**

Implementarea a fost realizata in OpenGL, cu ajutorulcaruia am folosit diverse functii, precum:

void mouseCallback(GLFWwindow\* window, double xpos, double ypos) – care face posibila miscarea prin scena cu ajutorul mouse-ului;

void processMovement() – care contine functionalitatile plasate pe taste;

bool initOpenGLWindow() – care initializeaza fereastra in OpenGL;

void initObjects() – care incarca obiectele in scena;

void initShaders() – care incarca shaderele in scena;

void drawObjects() – care deseneaza obiectele in scena;

void renderscene() – care are mai multe functionalitati, precum: incarca SkyBoxul, face animatiile, incarca lumina directionala, trimite toate datele la shadere, calculeaza toate aspectele importante pentru proiectarea scenei.

Cele mai multe modificari au fost facute in main.cpp, camera.cpp si in shadere. In camera.cpp se definesc miscarile inainte-inapoi-dreapta-stanga si se implementeaza glm::lookAt(cameraPosition, (cameraPosition + cameraFrontDirection), cameraUpDirection); pentru a functiona corect camera. De asemenea am adaugat o functie care imi printeaza coordonatele camerei void Camera::printPos() si doua functii pentru animatie void Camera::animatie() si void Camera::inainte().

In shaderStart.frag am implementat functiile pentru lumina, umbra, ceata si transparenta:

void computeLightComponents()

{ vec3 cameraPosEye = vec3(0.0f);

vec3 normalEye = normalize(fNormal);

vec3 lightDirN = normalize(lightDir);

vec3 viewDirN = normalize(cameraPosEye - fPosEye.xyz);

ambient = ambientStrength \* lightColor;

diffuse = max(dot(normalEye, lightDirN), 0.0f) \* lightColor;

vec3 reflection = reflect(-lightDirN, normalEye);

float specCoeff = pow(max(dot(viewDirN, reflection), 0.0f), shininess);

specular = specularStrength \* specCoeff \* lightColor;

lightDirN = normalize(torchPos-fPosEye.xyz);

float dist = length(torchPos-fPosEye.xyz);

float att = 1.0f / (constant + linear \* dist + quadratic \* (dist \* dist))\*10.0f;

ambient += att \* ambientStrength \* torchColor;

diffuse += att \* max(dot(normalEye, lightDirN), 0.0f) \* torchColor;

lightDirN = normalize(lampPos-fPosEye.xyz);

dist = length(lampPos-fPosEye.xyz);

att = 1.0f / (constant + linear \* dist + quadratic \* (dist \* dist))\*intensitate;

ambient += att \* ambientStrength \* lampColor;

diffuse += att \* max(dot(normalEye, lightDirN), 0.0f) \* lampColor;

lightDirN = normalize(lanternPos-fPosEye.xyz);

dist = length(lanternPos-fPosEye.xyz);

att = 1.0f / (constant + linear \* dist + quadratic \* (dist \* dist))\*intensitate;

ambient += att \* ambientStrength \* lanternColor;

diffuse += att \* max(dot(normalEye, lightDirN), 0.0f) \* lanternColor;

}

Umbra:

float computeShadow()

{ vec3 normalizedCoords = fragPosLightSpace.xyz / fragPosLightSpace.w;

normalizedCoords = normalizedCoords \* 0.5 + 0.5;

float closestDepth = texture(shadowMap, normalizedCoords.xy).r;

float currentDepth = normalizedCoords.z;

//float shadow = currentDepth > closestDepth ? 1.0 : 0.0;

float bias = max(0.05f \* (1.0f - dot(fNormal, lightDir)), 0.005f);

float shadow = currentDepth - bias > closestDepth ? 1.0 : 0.0;

if (normalizedCoords.z > 1.0f) {

return 0.0f; }

return shadow; }

Ceata:

float computeFog()

{ float fogDensity = 0.03f;

float fragmentDistance = length(fPosEye);

float fogFactor = exp(-pow(fragmentDistance \* fogDensity, 2));

return clamp(fogFactor, 0.0f, 1.0f); }

In main.cpp, cele mai multe modificari au fost facute in renderScene(), mai ales pentru animatii, si in processMovement(), pentru diferitele functionalitati pe care le am adaugat, cele mai importante fiin modurile de vizualizare: solid, wireframe si pointframe.

if (glfwGetKey(glWindow, GLFW\_KEY\_I))

{

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_POINT);

}

if (glfwGetKey(glWindow, GLFW\_KEY\_O))

{

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

}

if (glfwGetKey(glWindow, GLFW\_KEY\_P))

{

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

}

Skyboxul utilizat este acela din laborator, deoarece se potrivea cu scena implementata.

* 1. **Modelul grafic**

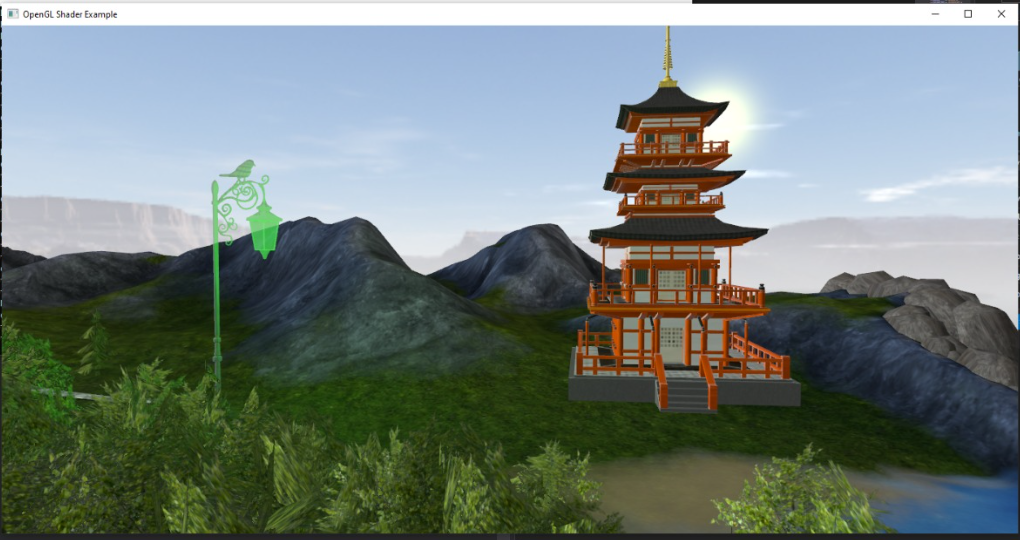
Scena si obiectele utilizate in proiect sunt Modele 3D care au fost aranjate cu ajutorul programului Blender si incarcate (exportate) cu extensia .obj in programul OpenGL scris in VisualStudio.

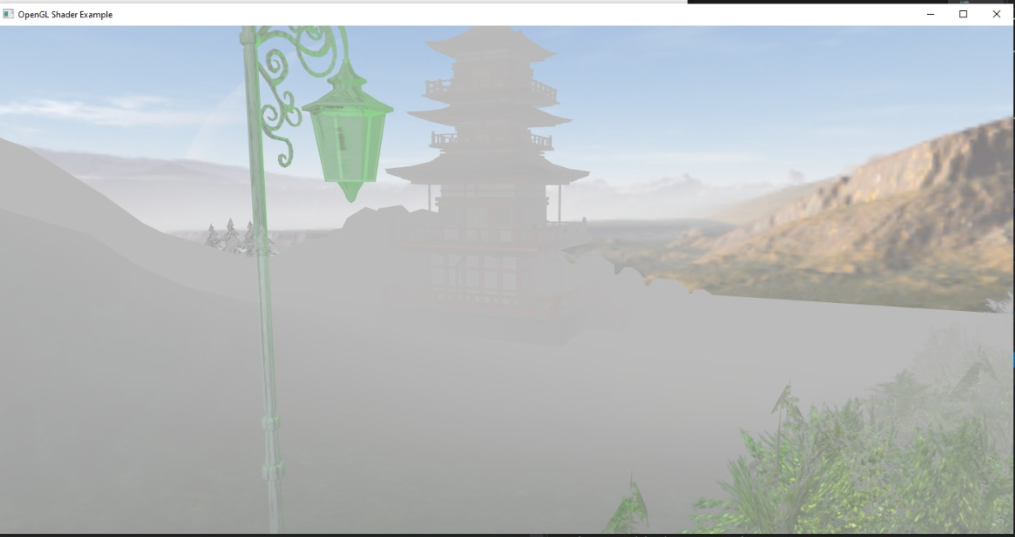
* 1. **Ierarhia de clase**

**Graphical user interface, application

Description automatically generated**

1. **Prezentarea interfetei grafice si manualul de utilizare**

****

****

****

**A picture containing nature

Description automatically generated**

**A picture containing sky, outdoor, aircraft, landing

Description automatically generated**

**A picture containing sky, water, outdoor, nature

Description automatically generated**

**A whiteboard with writing on it

Description automatically generated with low confidence**

**A picture containing map, text

Description automatically generated**

Utilizarea tastelor:

* M – harta de adancire
* W – deplasarea camerei in fata
* S – deplasarea camerei in spate
* A – deplasarea camerei in stanga
* D – deplasarea camerei in dreapta
* J,L – misca lumina directionala
* F,G – activeaza si dezactiveaza ceata
* I,O,P – modurile de vizualizare wireframe, solid si pointframe
* H – readuce avionul in pozitia initiala
* Z – animeaza cainele
* X – printeaza coordonatele camerei in pozitia in care te afli
* 0,1 – pornesc si opresc animatia scenei
* 2,3 – pornesc si opresc luminile punctiforme

1. **Concluzii**

In concluzie, acest proiect m-a invatat sa lucrez cu OpenGL si cu programul Blender, sa incarc si sa texturez obiecte, sa fac animatii, sa generez lumini si umbre si sa proiectez o scena grafica care este destul de realista.

1. **Bibliografie**

<https://moodle.cs.utcluj.ro/course/view.php?id=525>

<https://free3d.com/3d-model/old-lantern-pbr-98203.html>