MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA: Automatica si Calculatoare
SPECIALIZAREA: Calculatoare si Tehnologia Informatiei
DISCIPLINA: Prelucrare grafică
PROIECT: Future Desert OpenGL Scene

Prof. coordonator:

prof.ing. Nandra Cosmin

Student:

Gabriel Mihaila

Grupa: 30239

AN UNIVERSITAR

2021- 2022

Cuprins

2. Prezentarea temei alese

Obiectivul acestei proiect este de a sumariza cunostintele acumulate de-a lungul acestui semestru la disciplina Prelucrare Grafica, prin realizarea unei scene fotorealiste interactive.

Am ales sa implementez o scena artistica din desert, urmarind in principal calitatea texturarii imaginilor, aranjarea obiectelor in scena precum si calitatea randarii scenei (sa se apropie cat mai mult de realitate).

Complexitatea scenei a depins foarte mult atat de obiectele texturate pe care le contine, cat si de animatiile care ii confera un caracter interactive.

Folosind librariile OpenGL, GLFW si GLM, am reusit sa realizez urmatoarele aspecte:

- Intrarea in scena cu ajutorul unei animatii de prezentare;
- Vizualizarea scenei: scalare, translatie, rotatie, miscarea camerei, atat de la tastatura, cat si cu ajutorul mausului (la click + scrol);
- Vizualizarea scenei in modurile solid, wireframe, poligonal si smooth;
- Utilizarea luminii directionale pentru a lumina toata scena, precum si a unei lumini de tip spotlight, pentru a crea o atmosfera de seara in cadrul desertului;
- Maparea texturilor si definirea materialelor pentru un aspect cat mai placut al obiectelor
- Generarea umbrelor si a unui cub de lumina ce simuleaza miscarea soarelui de-a lungul unei zile (am incercat sa fac acest lucru cat mai realistic posibil, simuland apusul si rasaritul);
- Fotorealism, prin introducerea efectului de furtuna de nisip in scena, si, de asemenea, prin animarea diferitelor obiecte.

3. Scenariul

Scena se deschide printr-o animație care deschide imaginea de ansamblu a scenei in modul cinematografic, urmand ca mai apoi sa poata fi folosite tastatura si mouse-ul interactionad cu scena propriu-zisa.

3.1 Descrierea scenei si a obiectelor

Am creat o scena care cuprinde mai multe cadre suprinse intr-un desert muntos. Printre cadrele care fac parte din scena amintim: zona pieselor de sah in nisip (conceptual design), baza de cercetare si exploatare militara care include diverse cladiri si panouri solare, un humvee si un elicopter militaresc, precum si zona de petrol si alte deseuri chimice. O alta scena reprezinta zona de camping cu mai multi cactusi, zona de foc si corturile. O alta zona prezentata este oaza din cadrul desertului, precum si cea in care este prezentat Marele Sfinx, o piramida si mai multe camile.

Pentru a ne apropia cat mai mult de realitate, am incercat sa pastrez cat mai bine posibil dimensionalitatea obiectelor, precum si pozitionarea logica a acestora, formand niste asa-numite zone de interes (piesele de sah, zona de exploatare, camping-ul, oaza si zona cu piramide).

Aspectul ludic este conferit prin intrudcerea elementelor de tip stanca desertica, simuland ideea de canion desertic.

Pentru a spori si mai mult ideea de desert, am adaugat si un skybox care sa respecte aceasta tematica.

Exista si elemente de intregire a scenei, precum stancile amintite anterior, cactusii, lemnele pentru camping, precum si palmierii din jurul oazei.

Scena nu ar fi completă fara prezenta intr-un mod cat mai realistic posibil al muntilor, precum si simularea foarte apropiata de realitate a umbrelor mapate pe obiecte.

De asemenea, scena prezinta animatii ale masinii si ale elicopterului (elicopterul prezinta si detectiunea colizziunii cu pad-ul acestuia).

3.2 Functionalitati

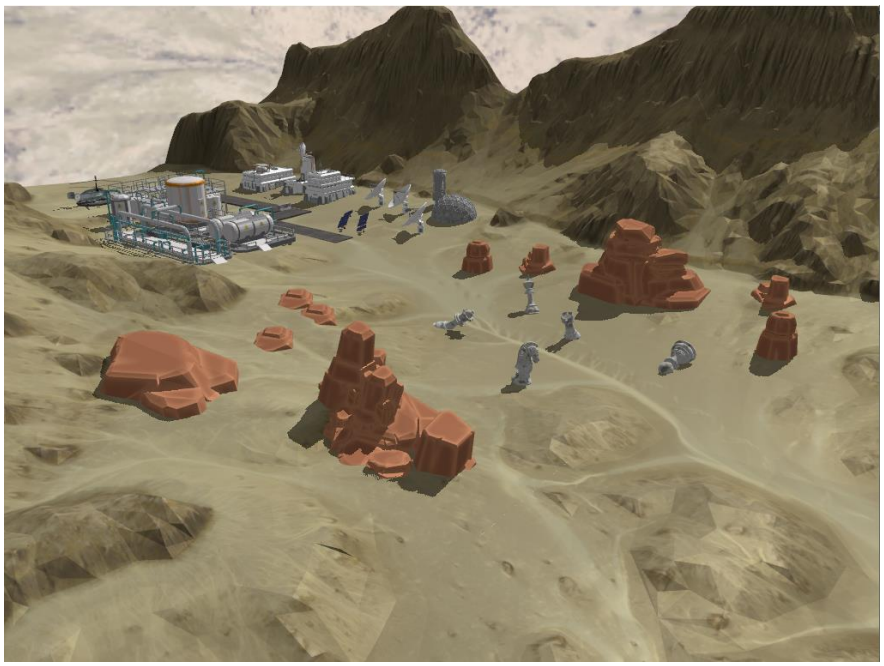
Utilizatorul se poate deplasa prin scena cu ajutorul mouse-ului si a tastelor **W, A, S, D.**

- Tasta W: **inaintare** in scena.
- Tasta A: deplasare **stanga**.
- Tasta D: deplasare **dreapta**.
- Tasta S: **departare** in scena.

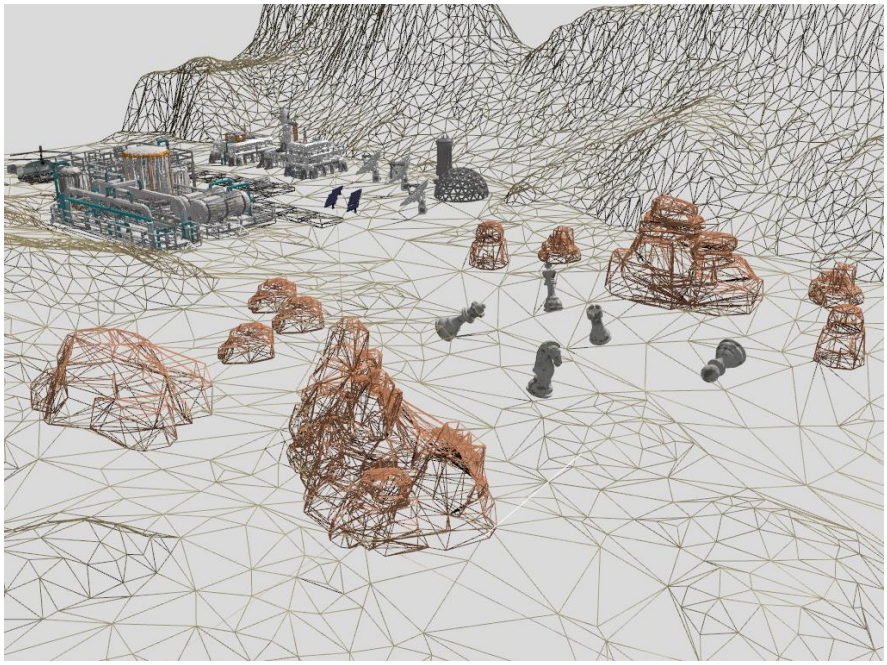
De asemenea, utilizatorul poate roti scena în sus, în jos, la stânga și la dreapta cu ajutorul mausului.

Utilizatorul poate alege sa vizualizeze scena utilizand un alt mod de prezentare. Aceste moduri se pot schimba cu ajutorul tastelor C, V, B care au urmatorul efect:

- Tasta C: vizualizarea scenei in modul de reprezentare **smooth/ plat.**



- Tasta V: vizualizarea scenei in modul de reprezentare **wireframe.**



- Tasta B: vizualizarea scenei in modul de reprezentare **punct.**

4. Detalii de implementare

4.1 Functii si algoritmi

Cel mai important pas in realizarea proiectului a fost reprezentat de aducerea obiectelor in scena, prin operatia initiala de incarcare a modelului 3D, urmand ca acesta sa treaca prin 2 etape de rendering: una pentru generarea umbrelor, iar cealalta pentru plasarea in scena.

In cod, aceste etape au fost implementate dupa cum urmeaza:
- incarcarea obiectului:

```
desertScene.LoadModel("objects/scena_v2/scena_v2.obj");
barreira.LoadModel("objects/barreira/barreira.obj");
humvee.LoadModel("objects/humvee/humvee.obj");
front_wheels.LoadModel("objects/front_wheels/front_wheels.obj");
back_wheels.LoadModel("objects/back_wheels/back_wheels.obj");
```

- functia de rendering a obiectului:

```
void renderScene(gps::Shader shader, bool depthPass)
{
    shader.useShaderProgram();

    model = glm::rotate(glm::mat4(1.0f), glm::radians(angleY), glm::vec3(0.0f, 1.0f, 0.0f))
    glUniformMatrix4fv(glGetUniformLocation(shader.shaderProgram, "model"), 1, GL_FALSE, gl

    //do not send the normal matrix if we are rendering in the depth map
    if (!depthPass) {
        normalMatrix = glm::mat3(glm::inverseTranspose(view * model));
        glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));
    }

    desertScene.Draw(shader);
}

void renderScene() {

    // depth maps creation pass
    //TODO - Send the light-space transformation matrix to the depth map creation shader and
    //       render the scene in the depth map

    depthMapShader.useShaderProgram();
    glUniformMatrix4fv(glGetUniformLocation(depthMapShader.shaderProgram, "lightSpaceTrMatrix"),
    glViewport(0, 0, SHADOW_WIDTH, SHADOW_HEIGHT);
    glBindFramebuffer(GL_FRAMEBUFFER, shadowMapFBO);
    glClear(GL_DEPTH_BUFFER_BIT);

    renderScene(depthMapShader, 1);
    renderBareira(depthMapShader, 1);
    renderHumvee(depthMapShader, 1);
    renderFront_wheels(depthMapShader, 1);
    renderBack_wheels(depthMapShader, 1);
    renderHelicopter_static(depthMapShader, 1);
    renderHelicopter_dinamic1(depthMapShader, 1);
    renderHelicopter_dinamic2(depthMapShader, 1);
    renderFish(depthMapShader, 1);
    glBindFramebuffer(GL_FRAMEBUFFER, 0);

if (showDepthMap) {
    glViewport(0, 0, (float)myWindow.getWindowDimensions().width , (float)myWindow.getWindowDimensions().height);

    glClear(GL_COLOR_BUFFER_BIT);

    screenQuadShader.useShaderProgram();

    //bind the depth map
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, depthMapTexture);
    glUniform1i(glGetUniformLocation(screenQuadShader.shaderProgram, "depthMap"), 0);

    glDisable(GL_DEPTH_TEST);
    screenQuad.Draw(screenQuadShader);
    glEnable(GL_DEPTH_TEST);
}
```

- apelul functiei de rendering, mai intai pentru a trece prin shader-ul ce contine harta de adancime specifica umbrelor, iar mai apoi prin shaderul obisnuit:

```
renderScene(depthMapShader, 1);
renderBareira(depthMapShader, 1);
renderHumvee(depthMapShader, 1);
renderFront_wheels(depthMapShader, 1);
renderBack_wheels(depthMapShader, 1);
renderHelicopter_static(depthMapShader, 1);
renderHelicopter_dinamic1(depthMapShader, 1);
renderHelicopter_dinamic2(depthMapShader, 1);
renderFish(depthMapShader, 1);
```

```
renderScene(myCustomShader, false);
renderBareira(myCustomShader, false);
renderHumvee(myCustomShader, false);
renderFront_wheels(myCustomShader, false);
renderBack_wheels(myCustomShader, false);
renderHelicopter_static(myCustomShader, false);
renderHelicopter_dinamic1(myCustomShader, false);
renderHelicopter_dinamic2(myCustomShader, false);
renderFish(myCustomShader, false);
```

Deoarece scena prezinta o zi futuristica dint-un desert montan, am vrut sa suprind cat mai bine fenomenele ce au loc de obicei aici, si anume furtuna de nisip, precum si nuanta luminii de tip spotlight resimitita din cauza reflexei luminii in particulele de nisip.

Pentru a activa/dezactiva efectul furtunii de nisip, se vor apasa tastele de F si G, din metoda **processMovement.**

```
if (pressedKeys[GLFW_KEY_F]) {
    bool_fog = 1;
    myCustomShader.useShaderProgram();
    fogLoc = glGetUniformLocation(myCustomShader.shaderProgram, "bool_fog");
    glUniform1i(fogLoc, bool_fog);
}

if (pressedKeys[GLFW_KEY_G])
{
    bool_fog = 0;
    myCustomShader.useShaderProgram();
    fogLoc = glGetUniformLocation(myCustomShader.shaderProgram, "bool_fog");
    glUniform1i(fogLoc, bool_fog);
}
```

Pentru a crea lumina de tip spot, am lucrat direct in "myBasicShader", trimitand comenzile de activare si dezactivare ale efectelor prin variabile boolene de tip uniform, a caror stare era modificata cu ajutorul tastelor.

Lumina tip spotlight, precum si efectul furtunii de nisip sunt create in fragment shader, dupa cum urmeaza:

```
float computeFog()
{
 float fogDensity = 0.05f;
 float fragmentDistance = length(fPosEye);
 float fogFactor = exp(-pow(fragmentDistance * fogDensity, 2));

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

void computeLightComponents()
{
```

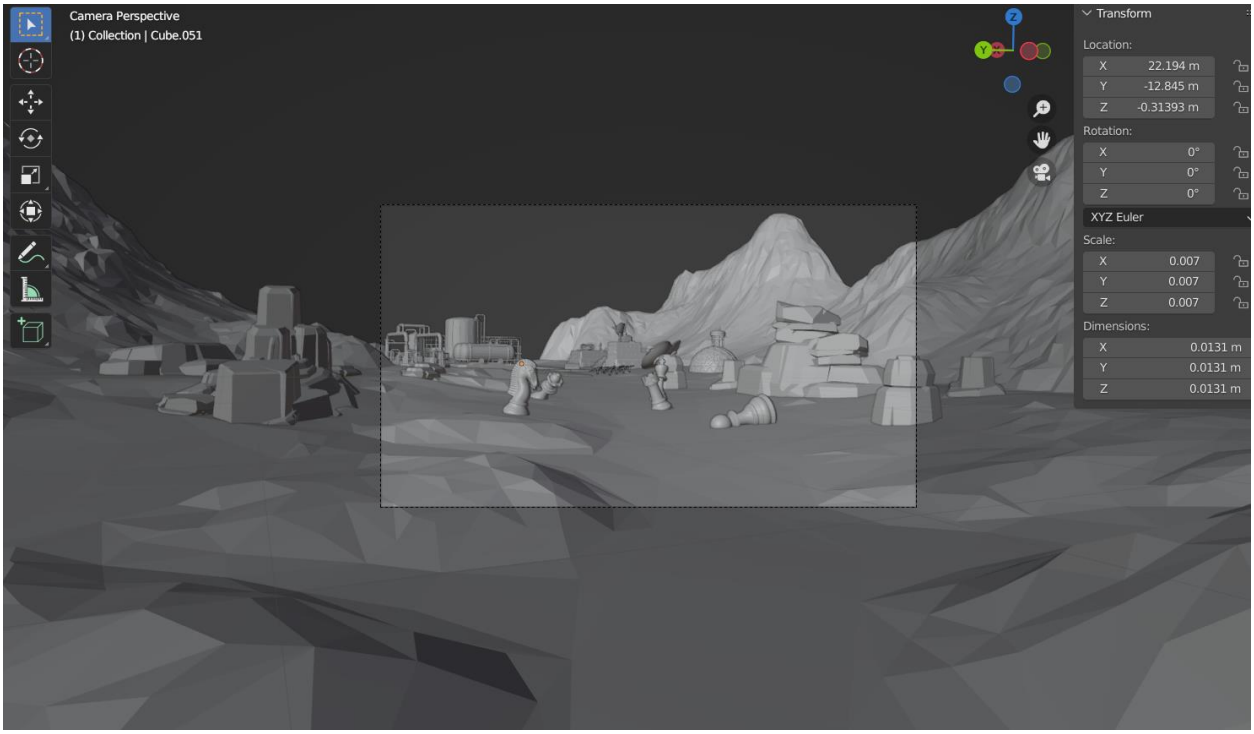
4.2 Modelul grafic

Modelele grafice sunt realizate pe baza modelelor geometrice care descriu forma si geometria obiectelor, insa cuprind si alte elemente ca textura, culoare, umbre, iluminare, necesare imbunatatirii realismului imaginii obiectului. Necesitatea utilizarii si altor elemente pentru model, in afara de modelul geometric, depinde de aplicatia pentru care a fost creat.

Obiectele alese pentru acest proiect au fost preluate de pe internet, urmand ca pozitionarea si texturarea lor sa fie realizate manual, cu ajutorul aplicatiei Blender.

Blender nu doar ca permite importul obiectelor si pozitionarea lor relativa, cat si crearea de modele grafice noi.

Un bun exemplu de obiect realizat 100% in Blender o reprezinta piramida, precum si drumul pe care este situata masina sau oaza.



4.3 Structuri de date

In cadrul proiectului, am utilizat structuri de date disponibile in biblioteca GLM, pentru functiile matematice folosite in cazul transformarilor geometrice, dar si structuri oferite de OpenGL, pentru rendering sau pentru comunicarea intre nivelul aplicatiei si cel al procesorului graphic (comunicarea cu shaderele).

Astfel, in functie de functionalitatea implementata, am utilizat urmatoarele structuri de date si tipuri speciale de variabile

- gps::SkyBox – crearea cubului infinit de jur imprejurul scenei

- gps::Camera – camera prin care se vizualizeaza scena

- GLuint – integer de tip uniform

- std::vector - vector pentru initializarea imaginilor din skybox

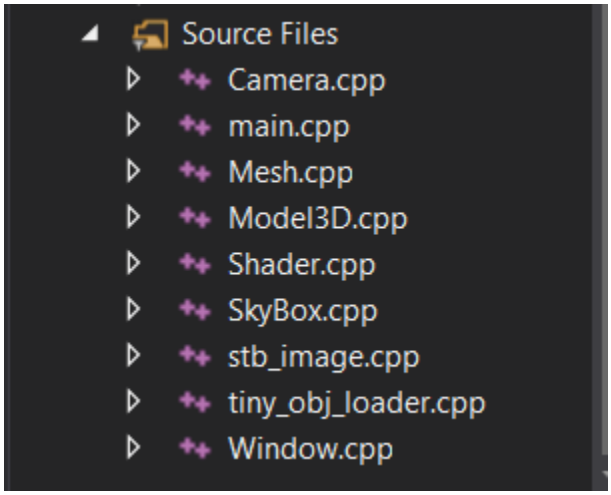
- glm::vec3 , mat 4 – pentru transformari geometrice

- gps::Shader – obiect de tip Shader pentru procesare in pipeline

- gps::Model3D – obiect pentru procesarea modelelor 3D

4.4 Ierarhia de clase

Libraria de clasa este atasata mai jos:



5. Prezentarea interfeței grafice utilizator / manual de utilizare

Controlul camerei se realizeaza cu tastele W,S,A,D, astfel:

- Tasta W: inaintarea in scena;
- Tasta A: deplasare stanga;
- Tasta D: deplasare dreapta;
- Tasta S: departare in scena;

Utilizatorul poate alege sa vizualizeze scena utilizand alt mod de prezentare. Aceste moduri se pot schimba cu ajutorul tastelor C, V, B care au urmatorul efect:

- Tasta C: vizualizarea scenei in modul de reprezentare smooth;
- Tasta V: vizualizarea scenei in modul de reprezentare wireframe;
- Tasta B: vizualizarea scenei in modul de reprezentare polygonal;

De asemenea, directia de deplasare poate fi influentata si de pozitia mouse-ului.

In continuare se va prezenta o legend a tastelor de pare se activeaza animatiile pe diferite obiecte si efecte:

- Tasta L: rotire la stanga a cubului de lumina

- Tasta J: rotire la dreapta a cubului de lumina
- Tasta G: activarea efectului de furtuna de nisip
- Tasta F: dezactivare efectului de furtuna de nisip
- Tasta 1: activare spotlight
- Tasta 2: dezactivare ceata
- Tasta I: animatie pestisor 1
- Tasta O: animatie pestisor 2
- Tasta UP: miscare masina fata
- Tasta DOWN: miscare masina spate
- Tasta Q, E: rotire scena de asmblu
- Tasta 5: ridicare elicopter de la sol
- Tasta 6: coborare elicopter la sol (implementata si coliziunea cu pad-ul)

6. Concluzii si dezvoltari ulterioare

In urma realizarii acestui proiect, am invatat sa creez o scena interactiva si sa modelez obiecte cu ajutorul software-urilor de proiectare. Am inteles mai aprofundat conceptele explicate la curs, incepand sa privesc obiectele din lumea inconjuratoare ca pe o posibila abstractizare in grafica computerizata.

Ca dezvoltari viitoare urmaresc: animatia elicopterului, efectul de foc in cadrul campingului, precum si efectul de night vision.

7. Referințe

- https://learnopengl.com/
- https://www.khronos.org/registry/OpenGL-Refpages/
- https://moodle.cs.utcluj.ro/course/view.php?id=418