**DOCUMENTAȚIE**

**Prelucrare grafică**

Documentație proiect OpenGL – Casa din poiană

**NUME STUDENT: Șchiop Adrian-Marian**

**Grupa: 30234**

An academic 2023-2024

Cuprins

1. Prezentarea temei
2. Scenariul
   1. Descrierea scenei și a obiectelor
   2. Funcționalități
3. Detalii de implementare
   1. Funcții și algoritmi
      1. Soluții posibile
      2. Motivarea abordării alese
   2. Modelul grafic
   3. Structuri de date
   4. Ierarhia de clase
4. Prezentarea interfeței grafice/ manual de utilizare
5. Concluzii și dezvoltări ulterioare
6. Referințe

**1.Prezentarea temei**

Scopul acestui proiect constă în realizarea unei scene 3D fotorealiste cu ajutorul bibliotecilor GLFW, GLEW, GLM etc. urmărind standardul OpenGL.

Scena proiectului întruchipează o fermă dintr-o poiană. În scenă există mai mulți arbori redând ideea de pădure, un drum către casă, un hambar și niște animale pe pășune.

Scena a fost creată cu ajutorul mediului de modelare 3D, Blender, iar partea de programare s-a realizat în IDE-ul VisualStudio.

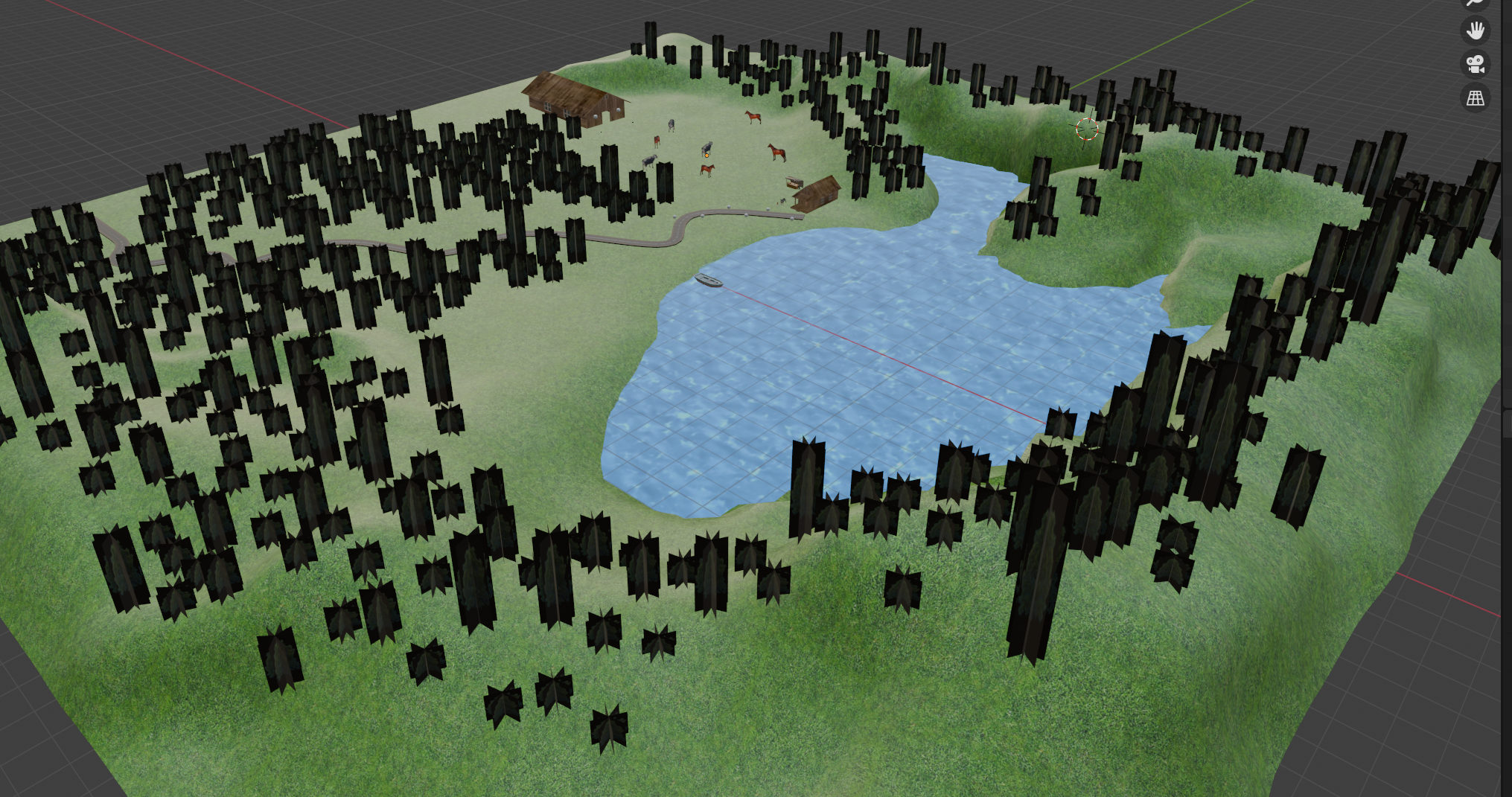
****

Fig1. Scena proiectului în Blender

**2.Scenariul**

**2.1 Descrierea scenei și a obiectelor**

Scena a fost realizată în Blender. Terenul a fost realizat manual, cu ajutorul instrumentelor puse la îndemână de către mediul de dezvoltare. Pe teren au fost importate diverse obiecte 3D cu extensia „.obj” precum: case, felinare, animale, un stand cu fructe, o barcă și un ceainic. Aceste obiecte au fost apoi texturate cu ajutorul unor imagini pentru a ilustra fotorealismul. Scena prezintă o fermă cu animale într-o pădure, unde avem un lac și o barcă. Pădurea a fost realizată cu ajutorul tehnicii de fragment discarding, intersectând 4 plane și aplicând aceeași textură pe fiecare dintre ele. Această tehnică elimină fundalul imaginilor îmbunătățind aspectul vizual al scenei.

|  |  |
| --- | --- |
|  |  |
| Fig 2.1 Scena în Blender | Fig2.2 Scena în proiect |
|  |  |

**2.1 Funcționalități**

În cadrul scenei, se navighează cu ajutorul mouse-ului și a tastaturii. În proiect au fost implementate funcționalități precum: lumina direcțională, lumina punctiformă, mișcarea camerei și posibilitatea de navigare prin scenă, proiecția umbrelor în scenă, rotirea scenei, afișarea ceții, introducerea întregii scene într-un skyBox. Toate aceste funcționalități sunt exemplificate mai jos.

**3.Detalii de implementare**

**3.1 Funcții si algoritmi**

Majoritatea funcțiilor au fost implementate în timpul laboratorului. Acestea au descris o bună parte din proiect: windowResizeCallback, keyboardCallback, mouseCallback, processMovement, initModels, initFBOs, initUniforms, renderScene(), initSkyBox.

*Lumina globală și lumina punctiformă*

În cadrul proiectului am implementat doua tipuri de lumini, o lumină globală si 9 lumini punctiforme. Pentru lumina globală am implementat funcția computeLightComponents utilizând modelul Phong. Pentru acest model avem nevoie de componentele ambientale, difuze și speculare și de poziția observatorului. La calculul componentei speculare, contează poziția observatorului(camera): float specCoeff = pow(max(dot(viewDirN, reflection), 0.0f), shininess).

Pentru implementarea luminii punctiforme am implementat funcția

computePointLight. Pentru calculul lumii punctiforme avem nevoie de 3 parametrii( constant, linear si quadratic) ce sunt transmiși drept variabile uniform din funcția „main” a programlui. Pentru fiecare lumină acești parametri sunt transmiși și sunt folosiți în calcularea atenuării.

float attenuation2 = 1.0 / (pointLightConstant + pointLightLinear \* distance2 + pointLightQuadratic \* distance2 \* distance2);

|  |  |
| --- | --- |
| Fig3.1 Lumină globală + lumină punctiformă | Fig3.2 Lumină punctiformă |

*Mișcarea camerei*

Pentru navigarea în scenă se utilizează mouse-ul și tastatura. Pentru navigarea în scenă cu ajutorul tastaturii a fost implementată funcția Camera::move din fișierul „Camera.cpp”, ce primește parametru direcția de mișcare a camerei și viteza de deplasare. Mișcarea cu ajutorul mouse-ului se realizează cu ajutorul funcției mouseCallback(GLFWwindow\* window, double xpos, double ypos)

În această funcție se preiau coordonatele inițiale ale mouse-ului pentru a se realiza rotația în jurul acestora

*Ceața*

Pentru implementarea ceții, am creat funcția creareCeata() unde am calculat un factor de ceață cu ajutorul funcției exponențiale ceataFactor = exp(-pow(fragmentDistance \* ceata.x, 2)).

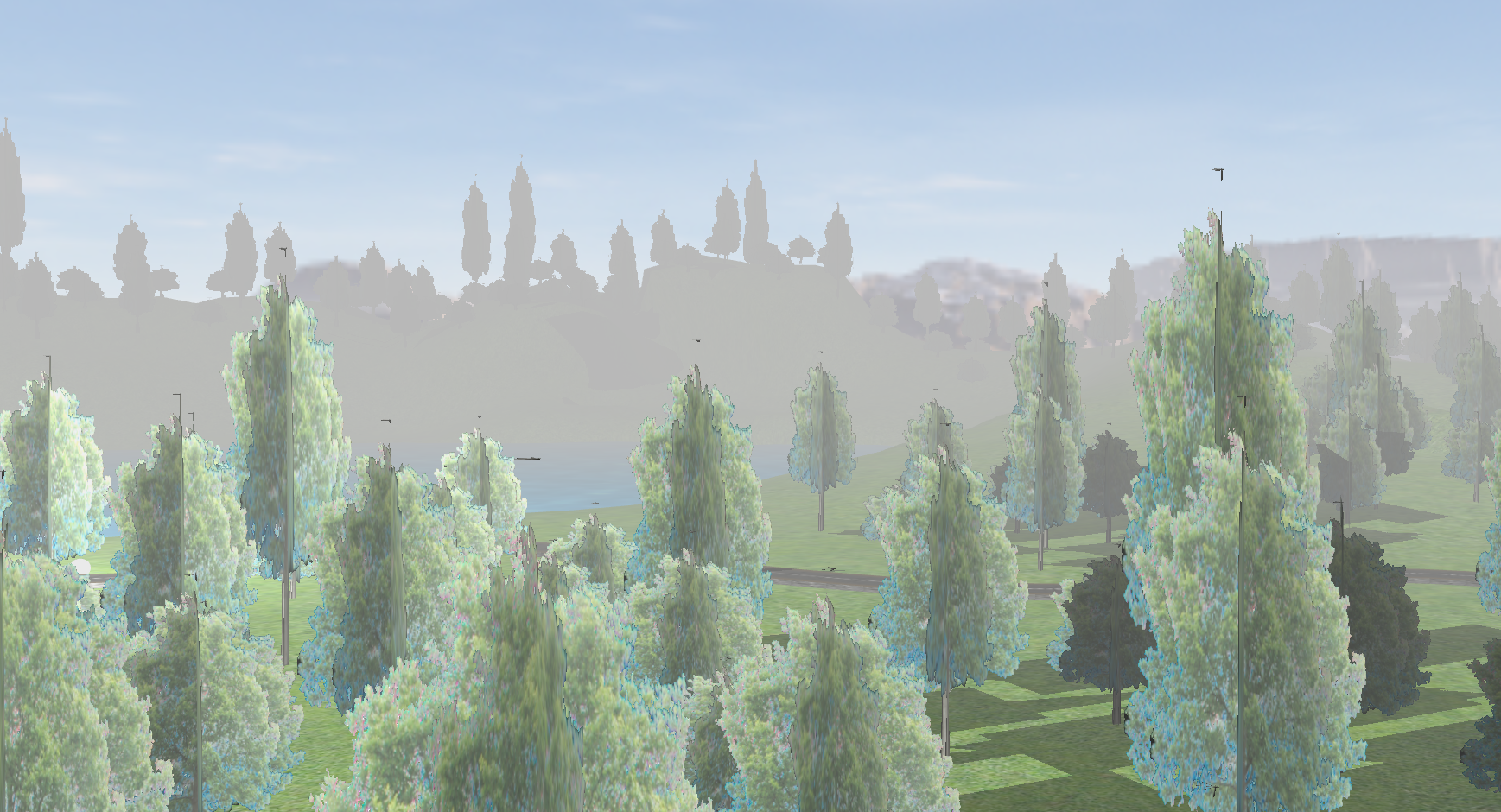


Fig3.3 Implementarea ceții

**3.1.1 Soluții posibile**

Anumite funcționalități din proiect pot fi realizate prin alte metode. Un exemplu elocvent îl constituie calculul lumii globale prin modelul de iluminare Blinn-Phong sau redarea efectului de ceață cu ajutorul unei ecuații pătratice sau liniare.

**3.1.2 Motivarea abordării alese**

Am optat în acest proiect pe calcularea luminii globale cu ajutorul modelului Phong deoarece acest model implementează mai veridic reflexiile speculare. De asemenea, am ales să redau efectul de ceață cu ajutorul funcției exponențiale pentru a ilustra mai bine fotorealismul.

**3.2 Modelul grafic**

Toate obiectele din scenă au fost modelate cu ajutorul mediului de dezvoltare Blender. Obiectele 3D au fost descărcate de pe anumite site-uri de specialitate. Relieful scenei a fost realizat de către mine, cu ajutorul instrumentelor de modelare puse la dispoziție. Obeictele au fost poziționate în blender la poziția pe care o doream în proiect, iar mai apoi exportate.

**3.3 Structuri de date**

Structurile de date folosite au fost: vec3, vec4, mat3, mat4, bool, float, GLint, etc. S-au folosit obiecte ce aparțineau claselor: Camera, Model3D, Mesh, Shader.

**3.4 Ierarhia de clase**

S-au folosit clasele “Camera” (operatii pe camera de vizualizat), “Mesh” (pentru aplicarea texturilor), “Model3D” (pentru operatiile cu obiecte), “Shader” (pentru utilizarea programelor de tip “shader”).

**4.** **Prezentarea interfeței grafice/ manual de utilizare**

Vizualizarea scenei cu ajutorul tastaturii:

* W – mișcare camerei înainte(efect de zoom-in)
* A – mișcare camerei spre stânga
* S - mișcarea camerei spre înapoi( efect de zoom-out)
* D – mișcarea camerei spre dreapta.

Vizualizarea scenei prin intermediul mouse-ului: utilizatorul va mișca de mouse, iar direcția de vizualizare a scenei se va schimba.

Accesarea diverselor funcționalități:

* Q – rotirea scenei spre stânga
* E - rotirea scenei spre dreapta
* K – rotirea luminii spre stânga
* L - rotirea luminii spre dreapta
* , - vizualizare scenă în mod wireframe
* . – vizualizare scenă în mod punctiform
* / - vizualizare scenă în mod solid
* O – vizualizare scenă în mod smooth

Pentru următoarele funcționalități trebuie să mișcați mouse-ul și să apăsați următoarele taste:

* I - oprirea luminii globale
* U – reactivarea luminii globale
* F – activarea ceții
* G - dezactivarea ceții

**5. Concluzii și dezvoltări ulterioare**

Acest proiect și-a atins scopul de familiarizare cu principiile de bază din OpenGL și ci software-ul de modelare grafica Blender.

Posibile dezvoltări ulterioare ar putea consta în implementarea animațiilor la apăsarea diverselor taste, interacționarea cu obiectele din scenă, de exemplu, mișcarea bărcii după o traiectorie dorită, implementarea coliziunilor

**6. Referințe**

1. <https://learnopengl.com/Getting-started/OpenGL>
2. <https://moodle.cs.utcluj.ro/course/view.php?id=612>
3. <https://docs.google.com/document/d/1njtWPMmOQNIaD_z9ve8iPRUqQTWdIV_PO-NvPD0nOuM/edit>
4. <https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM>
5. <https://stackoverflow.com/questions/8509051/is-discard-bad-for-program-performance-in-opengl>
6. <https://sketchfab.com/feed>
7. <https://free3d.com/>
8. <https://www.turbosquid.com/>