

Departmentul Calculatoare
Universitatea Tehnica din Cluj-Napoca



Proiectare grafica

Proiect

Nume: Varvara Raluca Ana-Maria
Grupa: 30232

Teaching Assistant: Alexandru Anisorac



Contents

1	Prezentarea temei	3
2	Scenariul	4
2.1	Descrierea scenei și obiectelor	4
2.2	Functionalități	4
3	Detalii de implementare	6
3.1	Functii si algoritmi	6
3.1.1	Solutii posibile	8
3.1.2	Motivarea abordării alese	9
3.2	Modelul grafic	9
3.3	Structuri de date	9
3.4	Ierarhia de clase	10
4	Prezentarea interfetei grafice utilizator/manual de utilizare	11
5	Concluzii si dezvoltari ulterioare	12

Chapter 1

Prezentarea temei

Tema acestui proiect constă în realizarea unei scene 3D, cât mai realistă și apropiată de detaliile vietii de zi cu zi. Scena pe care am ales să o modelez este o strada italiană cu coruri de iluminare stradală cu case și blocuri cu arhitecturi specifice, terase, mici parculete și personaje, atât pe timp de zi, putând intervenii ceata specifică climatului umed, cât și noaptea.

Scopul acestui proiect a fost învățarea principiilor din OpenGL și familiarizarea cu utilizarea practică a acestora în mediul de programare Visual Studio, utilizând librăriile prezentate la laborator (OpenGL, GLFW, GLM, etc.), de asemenea utilizarea mediului de modelare 3D Blender. Utilizatorul trebuie să aibă posibilitatea de a controla scena prin intermediul mausului și tastaturii.



Chapter 2

Scenariul

2.1 Descrierea scenei și obiectelor

Scena a fost construită cu ajutorul software-ului grafic "Blender" și cu ajutorul unor modele 3D cu extensia ".obj" preluate de pe diferite site-uri care propun diverse modele 3D. Aceasta este relativ simplă, conținând un număr mediu de obiecte, prezentând organizarea unei străzi italiene, cu case și blocuri pe fiecare parte a străzii, existând clădirea banchii, obiecte de iluminare stradală și banchi pe trotuar, o mașină care merge în față și în spate pe strada, o casă cu o curte cu copaci și o fântână, personaje și animale pe strada și un mic parculeț. Am modelat singura și o terasă prezenta în Fig1.

Figure 2.1: terasa



2.2 Functionalități

În cadrul scenei există mai multe posibilități de navigare: navigarea prin intermediul mouse-ului, prin intermediul tastaturii. Au mai fost implementate funcționalitățile cheie pentru un realism sporit, lumini direcționale și punctiforme, precum și câteva seturi de shadere (pentru obiecte, pentru crearea hărții de adâncime după care vor fi desenate umbrele, pentru SkyBox, etc). În plus, o altă funcționalitate este prezenta unei animații în scena, cu o mașină care se mișcă înainte și înapoi pe strada.

De asemenea, mai există funcționalități modificabile prin tastatura, precum pornirea unei cete care crește în timp, și oprirea ei care se face decremental tot în timp, vizual creându-se iluzia de lăsare a căii și de ridicare a acesteia, modificarea intensității luminilor corpurilor de iluminare stradală, modificarea la noapte și înapoi la zi. Prin intermediul apăsării diferitelor butoane ale tastaturii scena va putea fi vizualizată în mai multe moduri: în modul solid, wireframe și punctiform.

Figure 2.2:



Figure 2.3: Pisica care fugă de camera



Chapter 3

Detalii de implementare

3.1 Functii si algoritmi

In primul rand, avem functiile implementate pe parcursul laboratorului: *windowResizeCallback*, *keyboardCallback*, *mouseCallback*, *processMovement*, *initOpenGLWindow*, *initOpenGLState*, *initObjects*, *initShaders*, *initUniforms*, *initFBO*, *computeLightSpaceTrMatrix*, *drawObjects*, *renderScene cleanup*.

Figure 3.1:



Ceata

Pentru realizarea efectului de ceată au fost propuse în laborator mai multe tipuri de ceată: ceată lineară, ceată exponentielle și ceată exponentielle pătratică. Pentru acest proiect am abordat implementarea celei exponențiale pătratice. Factorul de ceată este procesat în funcția "computeFog()" din fragment shader cu ajutorul formulei:

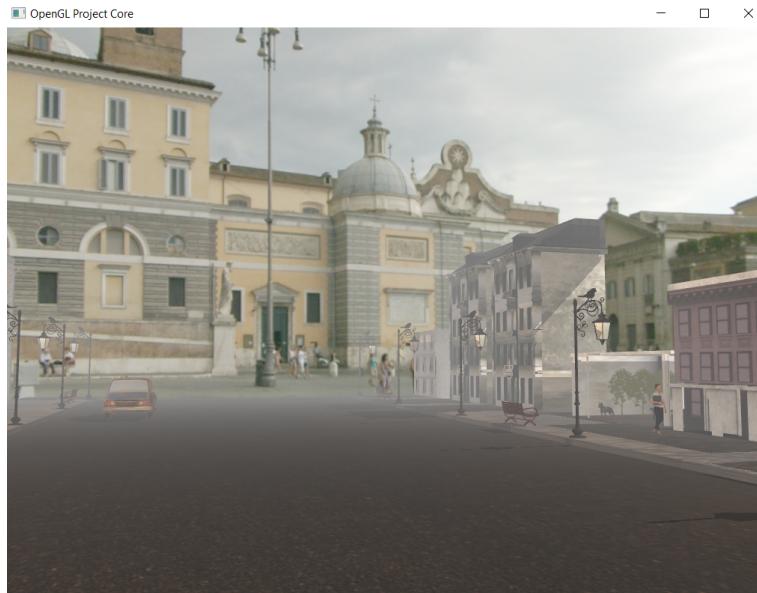
$$fog factor = e^{-(fragmentDistance*fogDensity)^2}$$

În vertex shader a fost calculată poziția obiectului și distanța relativă la camera de vizualizare, iar mai apoi s-a înlocuit în formula de mai sus.

Densitatea cetei se incrementează în timp până la o anumită valoare atunci când apasăm pe tastă f, iar după aceea ajunge la acea valoare, la apasarea tastei f din nou aceasta va scăde în

timp. Aceasta functie de crestere si scadere in timp a factorului de ceata este asemanator cu schimbarea factorului de translatie a masinii prezentat la animatie, mai jos.

Figure 3.2: Ceata



Lumina globală și lumina punctiformă

Pentru cele două surse de lumină am ales să implementez o lumină globală direcțională și o lumină de tip punctiform, care provine de la cele 20 de felinare de pe strada.

Pentru lumina globală, în cadrul fragment shader-ului există funcția ”computeLightComponents()”. Pentru această lumină sunt folosite componente: ambientală, difuză și speculară, calculate fiecare individual. Acest tip de lumină este calculat ținând cont și de poziția observatorului(a camerei) care contează pentru componenta speculară. Lumina ambientală , nu vine dintr-o anumita directie, este aproximarea luminii globale imprastiate în jurul scenei și reprezintă zona umbrila a obiectului. Lumina difuză este cea imprastiată în toate direcțiile de o sursă de lumina, intensitatea luminii fiind mai puternica pe fețele orientate direct către sursa de lumina (deci contează normala în punctul respectiv/ fragmentul deoarece folosim modelul de iluminare Phong, și directia sursei de lumina), iar cea speculară este reflectată direct de suprafața și se referă la cat de asemănatoare este suprafața cu o oglinda, deci depinde și de ”reflection”, pentru a îmbunătății efectul de iluminare.

Pentru lumina punctiformă, în cadrul fragment shader-ului există funcția: ”CalcPointLight()”. Pentru realizarea acestui tip de lumină a fost nevoie de 3 parametrii (constant, linear și quadratic) preluati din main și transmiși funcției din fragment shader și de un vector care să specifică poziția luminilor. Cu cât cei 3 parametrii au o valoare mai mică, cu atât intensitatea luminii va crește. Aceste valori pot fi modificate din tastatura și lăsa impresia cresterii intensității luminii. Factorul constant este lăsat la valoarea 1, și se modifică numai linear și quadratic, care sunt initializați cu 0.35 și 0.44. Atenuarea se calculează astfel:

$$\text{Attenuation} = 1 / (\text{Constant} + \text{Linear} * \text{Distance} + \text{Quadratic} * \text{Distance}^2)$$

iar acest coeficient va fi inmultit cu fiecare componentă: ambientală, difuză și speculară.

Deoarece avem 20 de lumini punctiforme cu 20 de poziții diferite, vom parcurge acestea cu un for și vom calcula distanța până la fragmentul procesat pentru fiecare și atenuarea pentru fiecare lampadar, și vom aduna fiecare componentă la Ambiental, difuz și specular.

Figure 3.3: Noapte + lumini punctiforme



Mișcarea camerei

Pentru explorarea scenei sunt folosite mouse-ul și tastatura. Rotația camerei se realizează cu ajutorul mouse-ului, iar mișările la stânga sau la dreapta, sau cele de fata, spate cu ajutorul tastaturii. Pentru rotația camerei cu ajutorul mouse-ului există în main funcția ”mouseCallback” care preia coordonatele inițiale ale mouse-ului, iar mai apoi realizează rotația camerei în funcție de mișările mouse-ului, coordonatele pointer-ului acestuia. Pentru navigarea în scenă cu ajutorul tastaturii a fost implementată în ”Camera.cpp” funcția ”move” care primește ca și parametru direcția de mișcare a camerei și viteza cu care aceasta să se deplaseze în direcția indicată de primul parametru. În interiorul acestei funcții, cu ajutorul acestor doi parametrii se vor modifica parametrii camerei: ”cameraPosition” și ”cameraTarget”, în funcție de direcția în care se dorește deplasarea camerei.

Mișcarea masinii

Am implementat și o animație cu o mașină care se mișcă pe strada, aceasta este importată ca un obiect diferit fata de scena și ne folosim de faptul că plasam obiectele în spațiu global cu ajutorul matricei de model. Astfel înmulțim la matricea de model deja existentă o matrice de translație cu un factor pe axa x care crește sau scade în timp (a mai fost nevoie să dețină o matrice de rotație în jurul axei y deoarece mașina era amplasată perpendicular pe strada). Mașina se poate deplasa în față sau în spate, și modurile se schimbă atunci când mașina a ajuns la maximul sau minimul (adică marginile străzii) la care se poate deplasa. Avem o funcție care calculează factorul cu care se modifică pe axa x translația mașinii cu ajutorul timpului pe care îl luăm cu `glfwGetTime`, avem 2 variabile de timp: `old` și `new`, și prin diferența dintre aceste 2 variabile înmulțim cu viteza cu care se mișcă mașina adăugăm sau scadem în funcție de modul în care se mișcă în față sau în spate la factorul de translație.

3.1.1 Solutii posibile

Soluțiile posibile sunt cele prezentate și explicate mai sus, există și alte soluții posibile, de exemplu pentru calcularea coeficientilor ambiental, difuz și specular la lumini se putea calcula

cu ajutorul modelului de iluminare Blinn-Phong, sau ceata se putea implementa cu un alt algoritm.

3.1.2 Motivarea abordării alese

Am ales implementarile si algoritmii de mai sus pentru mai multe motive, am ales Phong in loc de Blinn Phong deoarece cel din urma creeaza o implementare mai bune pentru reflexiile speculare, iar in scena mea nu exista obiecte care au materiale care seamana cu o oglinda, deci nu ar fi fost ceva vizibil imbunatatit pentru fotorealismul scenei.

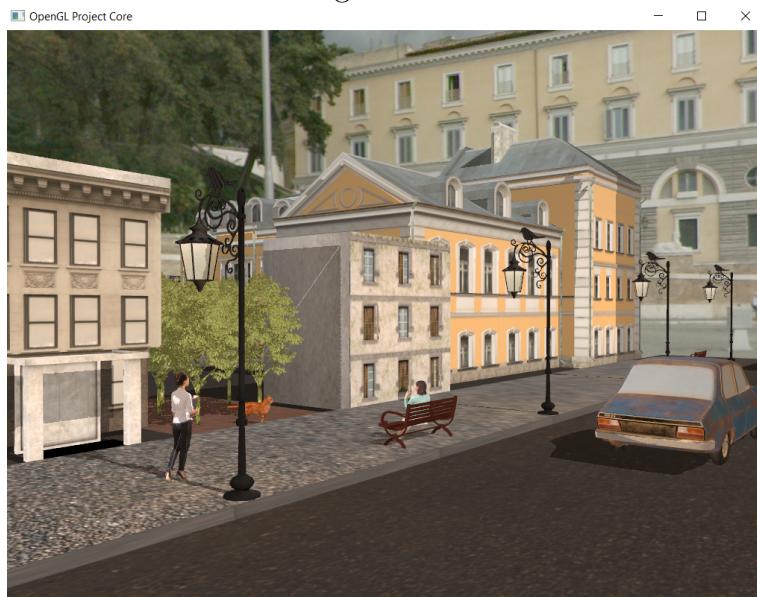
3.2 Modelul grafic

Toate obiectele au fost editate cu ajutorul programului Blender. Majoritatea obiectelor au fost descarcate si importate si pozitionate in Blender iar altele au fost create de mine, precum strada, trotuarul si mica terasa. Trotuarul a fost un plan la care am mapat o textura potrivita, iar trotuarul este un set de cuburi scalate si mapate cu o textura potrivita. Terasa a fost creeata cu o combinatie de cuburi scalate si alte parti selectate de la alte obiecte (acoperisul casei din stanga).

Avem 21 de surse de lumina, dintre care o sursa directionala care reprezinta soarele, si alte 20 care reprezinta lumina de la corpurile de iluminare stradala.

Umbrele sunt desenate intr-o textură 2D, de adancime, care este apoi folosită pentru desenarea acestora in scena finală.

Figure 3.4:

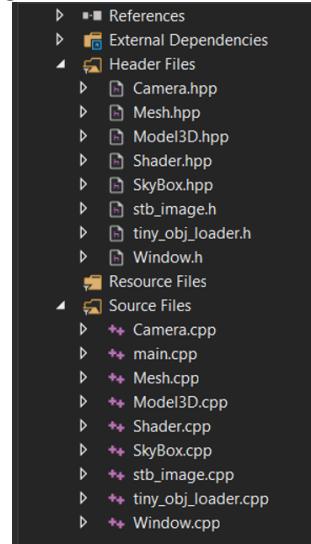


3.3 Structuri de date

Nu au fost implementate structure de date aditionale sau clase in plus fata de template, au fost folosite doar structurile de date oferite de bibliotecile folosite (vec3, vec4, mat3, mat4 etc.).

3.4 Ierarquia de clase

Figure 3.5: ierarquia de clase



Chapter 4

Prezentarea interfetei grafice utilizator/manual de utilizare

Pentru navigarea în scenă utilizatorul poate să aleagă din următoarele metode:

1. Vizualizarea scenei cu ajutorul tastaturii:

- "A" va deplasa camera în direcția stângă
- "D" va deplasa camera în direcția dreaptă
- "W" va deplasa camera în direcția înainte(efect de zoom-in)
- "S" va deplasa camera în direcția înapoi(efect de zoom-out)

2. Vizualizarea scenei prin intermediul mouse-ului: mișcarea mouse-ului înainte și înapoi va deplasa camera în sus și în jos, schimbând direcția de privire a scenei, iar în stanga și în dreapta va roti camera în jurul axei y pentru a vizualiza scena în toate partile.

Mai sunt și alte taste care au diverse funcționalități precum:

- "1" pentru vizualizarea scenei în modul solid
- "2" pentru vizualizarea scenei în modul wireframe
- "3" pentru vizualizarea scenei în modul punctiform
- "I" pentru marirea intensității luminii felinarelor
- "K" pentru micsorarea intensității luminii felinarelor
- "N" pentru mutarea în modul noapte
- "Z" pentru revenirea la modul zi
- "F" pentru pornirea sau oprirea cetei, ceata poate fi oprită numai cand factorul de densitate al cetei a ajuns peste un anumit threshold pentru a arata tot efectul de ceata.
- "V" pentru pornirea animatiei camerei
- "M" pentru pornirea muzicii

Chapter 5

Concluzii si dezvoltari ulterioare

Proiectul a fost un mod bun si creativ de familiarizare cu principiile OpenGL, dar si cu software-ul grafic 3D "Blender" in care s-a dezvoltat scena initial. Opengl prezinta un pipeline grafic destul de puternic pentru dezvoltarea de aplicatii 2D/3D.

Alte dezvoltari ulterioare ar fi implementarea modelului de iluminare Blinn-Phong si poate si Ray tracing pentru un model de iluminare mai realist. S-ar putea implementa intrarea in case sau in blocuri si interactiunea cu paharele sic anile de cafea de pe terasa, sau mai multa interacciune cu animalutele si cu personajele. S-ar putea implementa si lasarea serii in timp, succesiv.

Bibliography

<https://learnopengl.com/Introduction>

<https://free3d.com/>

<https://free3d.com/>

<https://www.cgtrader.com/free-3d-models>

<https://www.turbosquid.com/Search/3D-Models/free>

<https://learnopengl.com/Lighting/Multiple-lights>

<https://community.khronos.org/t/passing-array-of-vec3-to-fragment-shader/74450>

<https://www.textures.com/library>

