**Universitatea Tehnică din Cluj-Napoca**

Departamentul Calculatoare

**Prelucrare Grafică:**

**Proiect OpenGL**

Student: Copoț Raluca

Grupa: 30237

Data: 13.01.2022

Cuprins

**Prezentarea temei3**

**Scenariul3**

Descrierea scenei si a obiectelor3

Functionalitati5

**Detalii de implementare5**

Functii si algoritmi5

Modelul grafic6

Structuri de date6

Ierarhia de clase7

**Manual de utilizare7**

**Concluzii si dezvoltari ulterioare8**

**Referinte8**

1. **Prezentarea temei**

Tema acestui proiect este realizarea unei scene 3D, folosindu-ma de API-ul OpenGl, de mediul de dezvoltare Visual Studio si de limbajele de programare C/C++. Pentru a usura acesta sarcina, m-am folosit de bibliotecile: OpenGL Mathematics (GLM)[1], biblioteca de matematica C++; (GLFW)[2], care ofera un API pentru crearea ferestrelor si pentru receptionarea inputului si evenimentelor; (GLEW)[3], folosita pentru a determina versiunea optima de OpenGL.

Am folosit, de asemenea, notiunile invatate in acest semestru la laboratoarele de Prelucrare Grafica, pentru a realiza toate elementele prezente in acest proiect, precum posibilitatea de a naviga in lumea create, sau skybox-ul, pentru a crea iluzia unei lumi nelimitat de mare. De asemenea, pentru obiecte si texturiam folosit resurse de pe internet, care se afla in domeniul public.

1. **Scenariul**

**2.1. Descrierea scenei si a obiectelor**

Pentru modelarea scenei am fost inspirata de jocul video preferat, The Witcher 3: Wild Hunt, dezvoltat de CD Projekt RED, mai exact de zona Crookback Bog, Aceasta este o mlastina in care se afla „Ladies of the woods”. Astfel, am decis sa ii ofer lumii pe care am modelat-o aspectul unui mic sat, retras, misterios, in care ar putea trai vrajitoare sau alte creaturi magice.

Elementul principal este „satu” in sine, cateva casute care inconjoara o fantana, zona care este lumina de cateva lampi. Zona din care incepe explorarea este in fata unui pod, ce strabate un rau, pentru a ajunge in acest sat. De asemenea, in aceasi directie se afla o casa neobisnuita, cu picioare, care pare ca urmeaza sa se alature celolaltor casute. Am incercat, pe cat posibil, sa nu fac aceasta lume infricosatoare, dorind sa starnesc interes si curiozitate, nu frica sau senzatia de pericol.

Pentru a crea iluzia unei lumi locuite, fara a adauga oameni, am importat o pisica, si cateva flori. De asemenea, pentru a fi pe cat posibil apropiata de o padure, am adaugat copaci si ciuperci.

**O imagine care conține iarbă, arbore, exterior, bogat

Descriere generată automat**

O imagine care conține arbore, exterior

Descriere generată automat

**2.2 Functionalitati**

Pentru a naviga prin scena, utilizatorul poate folosi atat tastele W (inainte), A (stanga), S (inapoi), D (dreapta), space (sus), control (jos), cat si mouse-ul. Cu tastele Z si X se poate roti spre stanga, respectiv dreapta. Cu tastele 1, 2, 3, 4 modifica modul de vizualizare in wireframe, solid, smooth sau polygonal. Cu Q si cu E roteste intreaga scena, camera insa ramane in aceeasi pozitie.

Utilizatorul poate si sa interactioneze cu lumea in sine, astfel:

* O: creste densitatea cetii
* P: scade densitatea cetii
* J: roteste in sens trigonometric sursa de lumina directionala
* L: roteste in sensul acelor de ceasornic sursa de lumina directionala
* R: activeaza/dezactiveaza animatia creaturilor care zboara
* M: afiseaza shadow map-ul

1. **Detalii de implementare**

**3.1. Functii si algoritmi**

In primul rand, pentru a naviga prin scena ma folosesc de formulele date in laborator pentru a calcula vectorii si componentele corespunzatoare. Utilizez, in special, transformari ale camereisi functia lookAt, pentru a ma asigura de faptul ca voi putea orienta camera in directa corecta dupa fiecare mutare a utilizatorului.

Am implementat animatii pentru creaturile zburatoare, carora le voi pune fluturi in urmatoarele mentiuni din documentatie, si pentru cazanul de vrajitoare. In cazul fluturilor, efectuez o translate in origine, o rotatie cu un unghi delta, care se incrementeaza de fiecare data cand sunt desenate obiectele, si poi o translatie cu o matrice inversa, pentru a adduce obiectul inapoi. Astfel, fluturii se vor invartii in jurul podului. Pentru cazan, folosesc o variabila, numita delta2, pe care o incrementez sau decrementez in functie de niste valori limita. Aceasta valoare este folosita pentru translatii pe axa y.

Iluminarea este realizata utilizand modelul Phong. Pentru iluminarea directa folosesc exemplul prezentat in tutorialele din indrumatorul de laborator. Calculez pe rand componentele difuza, ambientala si speculara, le modulez cu texturile, iar la final, folosind si valoarea cetii, respective al umbrei, calculez culoarea fiecarui fragment cu formulele:

color = min((ambient + (1.0f - shadow)\*diffuse) + (1.0f - shadow)\*specular, 1.0f);

fColor = fogColor \* (1 - fogFactor) + vec4(color \* fogFactor, 0.0f);

fColor reprezinta culoarea finala a fragmentului.

Lumina punctiforma este, la randul ei, calculate cu ajutorul unei formule, asemanator cu cea directionala, insa ma folosesc de un factor de atenuare, care asigura faptul ca lumina va deveni din ce in ce mai putin puternica pe masur ace observatorul se va indeparta de sursa de lumina.

Umbrele sunt generate folosind tehnica de Shadow Mapping, cu un buffer de adancime. Etapele principale ale acestei tehnici sunt rasterizarea scenei din punctul de vedere al luminii, fiind stocata in adancimea (in harta de adancime). A doua etapa consta in rasterizarea scenei din punctul de vedere al observatorului, adica din pozitia camerei. Fragmentele care vor avea o adancime mai mare decat cea anterioare vor fi considerate „in umbra”.

Pentru ceata, utilizez formula pentru ceata exponentiala patratica. Modific apoi culoarea calculata anterior, cu acest factor de ceata. De asemenea, utilizez tehnica de eliminare a fragmentelor, aplicand functia discard pentru orice fragment care are o valoare mai mica de 0,1. Acest lucru este aplicat pentru textura copacilor.

**3.2. Modelul grafic**

Pentru aceasta scena 3D utilizez trei modele. Primul este scena in sine, care contine toate elementele statice. Celelalte doua sunt ceaunul, respectiv fluturii, care sunt animati folosind translatii, respectiv rotatii.

Intreg modelul grafic in sine poate fi mutat sau rotit, in functie de inputul dat de utilizator. De asemenea sunt utilizate iluminari si restul functilor si algoritmilor precizati anterior. Modelul este texturat, pentru a parea mai realist. Insa, pentru a putea rula programul si pentru a putea exporta scena, care a fost modelata in blender, am redus numarul de fete al obiectelor, implicit si calitatea texturilor. Acest lucru este vizibil in special in cazul casei cu picioare.Pentru implementare am utilizat un tutorial pentru OpenGl[4] , resursele de la laborator, dar si forumurile[5].

**3.3. Structuri de date**

Pentru implementarea acestui proiect am utilizat structurile de date specifice OpenGL, mai exact bibliotecii glm. In principal, am utilizat matrici de 3x3 sau 4x4, adica mat3, mat4 si vectori de doua, trei si patru elemente, adica vec2, vec3 si vec4. De asemenea, am folosit Glint, GLuint si GLfloat pentru diversi parametri, cum ar fi unghiul cu care rotesc lumina, sau pentru tipul vectorului in care stochez pozitiile luminii punctiforme.

**3.4. Ierarhia de clase**

Proiectul contine mai multe clase, de tipul cpp, pentru fiecare dintre ele fiind, de asemenea, definit un fisier de tip hpp. In fisierele de tipul hpp sunt dclarate clasele, respectiv metodele utilizate, iar in cpp-uri ele sunt implementate. In afara de clasa main, toate fisierele au un hpp atasat, ea continand obiecte de tipul toturor celorlaltor clase.

Camera contine metodele necesare pentru a face deplasarea posibila. Clasa Mesh initializeaza bufferele pentru toate obiectele incarcate in proiect. Folosindu-se de Mesh, clasa Model3D se ocupa de prelucrarea modelelor, pe care le incarca, le aplica texturile si carora le initializeaza buffer-ul de date. Clasa Shader incarca shaderele din proiect, le compileaza si realizeaza legatura intre fisierele .vert si cele .frag, trimitand datele in mod corespunzator. Skybox se ocupa cu incarcarea acestei componente, precum si clasa Window are ca rol operatiile cu fereastra. Clasa stb\_image contine functiile de baza pentru aplicatia OpenGL, iar clasa tiny\_obj\_loader este cea in care se afla metodele de baza pentru incarcarea obiectelor.

1. **Manual de utilizare**

Pentru a putea naviga in scena, utilizatorul se poate deplasa folosind tastele:

* W, A, S, D: pentru deplasarea in fata, la stanga, in spate, respectiv in dreapta
* Space, control: pentru deplasarea in sus, respectiv in jos
* Z, X: pentru a roti orientarea camerei spre stanga, respectiv spre dreapta

De asemenea, pentru a modifica mai usor orientarea camerei, utilizatorul poate utiliza mouse-ul, mutand cursorul in directia dorita.

Pentru a interactiona cu scena, utilizatorul poate folosi urmatoarele taste:

* Q, E: pentru a roti intreaga scena, insa pozitia utilizatorului (a camerei) ramane neschimbata
* 1, 2, 3, 4: pentru a schimba modul de vizualizare in wireframe, solid, smooth si polygonal
* O, P: pentru a creste, respectiv a diminua densitatea cetii
* J, L: pentru a roti sursa de lumina spre stanga, sau spre dreapta
* R: pentru a porni sau a opri animatia fluturilor
* M: pentru a afisa shadow map-ul si a vedea scena din perspectiva sursei de lumina.

In rest, utilizatorul poate explora si manipula intreaga scena dupa cum doreste.

1. **Concluzii si dezvoltari ulterioare**

Pentru a crea o scena realista 3D este necesar un efort mult mai mare decat pentru a crea una care poate incalca anumite reguli ale realismului. Desi am implementat umbre, aceste nu se aplica pentru intreaga scena, deoarece nu am reusit sa gasesc o valoare corespunzatoare pentru pozitia luminii. Acest lucru este vizibil in special in cazul im care este rotita, ea vazand numai jumatate de scena.

Un alt lucru pe care l-as imbunatati este adaugarea umbrelor si pentru sursele punctiforme de lumina. De asemenea, as cauta parametrii corespunzatori pentru a obtine efectul dorit, in prezent sursele fiind prea luminoase. Iar, pentru animatia scenei, am reusit doar sa modific pozitia finala, insa fara a vedea cum sunt efectuate mutarile efectiv, acesta fiind un alt lucru pe care l-as imbunatati.

1. **Referinte**

1 - <https://github.com/g-truc/glm>

2 - <https://www.glfw.org/>

3 - <http://glew.sourceforge.net/>

4 - <https://learnopengl.com/>

5 - <https://stackoverflow.com/>