Proiect SPG

# Cuprins

1. Prezentarea temei
2. Scenariul
   1. Descrierea scenei si a obietelor
   2. Functionalitati
3. Detalii de imlementare
   1. Functii si algoritmi
      1. Solitii posibile
      2. Motivarea abordarii alese
   2. Structuri de date
   3. Ierarhia de clase
4. Prezentarea interfetei grafice / manual de utilizare
5. Concluzii si dezvoltari ulterioare
6. Referinte

# Prezentarea temei

Proiectul propune realizarea unei scene 3D cu ajutorul librariei OpenGL. Pentru realizarea acestei scene ma voi folosi de o serie de obiecte 3D din aceeasi tema aranjate in scena, de mai multe surse de lumini, realizarea de umbre pentru obiecte si animatii simple.

# Scenariul

* 1. Descrierea scenei si a obiectelor

Scena realizata prezinta o imagine din seria Star Wars. Este vorba despre o cucerirea a unei planete de catre Imperiu si stabilirea trupelor necesare pentru garnizoana. In cadrul acestei scene se gasesc o serie de obiecte: un teren de desert, personaje Star Wars (Darth Vader, Darth Maul), un batalion de Stormtroop-eri, un dorid (BB8), mai multe nave de lupta Imperiale, doua feluri de tancuri armate pentru anihilearea trupelor terestre si cateva corpuri decedate.

Obiectele folosite pentru acest proiect se gasesc pe website-ul TF3DM (<http://tf3dm.com/>) si pot fi cautate utilizand in bara de Search tag-ul “star wars”.

Punerea in scena a acestor obiecte 3D a fost realizata cu ajutorul programului 3DS Max. Folosindu-ma de el am aranjat obiectele pe pozitiile dorite, am redimensionat obictele astfel incat marimea lor sa para naturala si am aplicat texture pe unele dintre obiectele amintite mai sus. Avantajele pe care mi le-a adus programul 3DS Max au fost urmatoarele: modelarea si texturarea relative usoara a obiectelor, diminuarea gradului de dificultate al aranjarii obiectelor in scena 3D, iar cel mai important, transmiterea pozitiei in scena a obietului prin optiunea de export. Astfel, nu a mai fost nevoie de folosirea operatiilor de translatie, rotatie sau scalare din aplicatia OpenGL.

* 1. Functionalitati

Pentru a interactiona cu scena 3D, utilizatorul se poate folosi de o serie de functionalitati care au fost implementate pe parcursul realizarii proiectului. Prima dintre acestea este deplasarea camerei prin scena utilizand tastele W (fata), A (stanga), S (spate), D (dreapta). Mai apoi, a fost adaugata o functionalitate pentru rotirea camerei utilizand mouse-ul si comutatoare pentru sursele de lumina (tastele 1 si 2). Proiectul dispune si de o functie de animatie de prezentare (asemanator unui bench-test grafic), care poate fi pornita si oprita de pe tastele O si respectiv P. Ultima functionalitate este cea de vizualizare a scenei. Aceasta poate fi vizualizata in modurile solid (tasta 7), wireframe (tasta 8) si polygonal (tasta 9).

# Detalii de implementare

* 1. Functii si algoritmi
     1. Solutii posibile
* Folosirea tehnicii “CubeMap”

Aceasta tehnica nu reprezinta altceva decat incadrarea scenei de obiecte 3D in interiorul unui cub. Totodata, tehnica de “CubeMap” elimina muchiile cubului, oferind utilizatorului iluzia unei sfere. Fetele cubului reprezinta imagini de fundal, pozitionate la infinit, ceea ce inseamna ca nu se poate trece cu camera prin peretii cubului. Fiecare fata a cubului poate fi texturata. Pentru acest proiect, am ales texturarea fetelor cubului cu imagini ale une planete extratereste.

* Sursele de lumina

Lumina in OpenGL este o iluzie. Senzatia de lumina se realizeaza prin modificarea culorii unoi obiect, astfel incat acesta sa dea impresia ca ar fi luminat, sau ca reflect alumina.

Exista mai multe tehnici pentru implementarea surselor de lumina. Pentru implementarea surselor de lumina in acest proiect am folosit tehnica Phong. Aceasta tehnica considera ca lumina reflectata de un obiect este constituita din trei componente: lumina ambientala, lumina difuza si lumina speculara.

Lumina ambientala nu vine dintr-o directive specifica, afectand toata scena de obiecte 3D. Aceasta lumina nu depinde de pozitia viewr-ului, sau de directia luminii.

Lumina difuza este impartita egal in toate directiile. Existenta luminii difuze pe un suprafata nu depinde de pozitia viewer-ului, dar depinde de directia luminii. Lumina este mai intense pe suprafetele care sunt orientate direct spree a si isi pierde din intensitatea atunci cand obiectele nu sunt indreptate spre ea. Calculul acestei lumini depinde de normal suprafetei si directia luminii.

Lumina speculara este lumina reflectata direct de catre suprafata si se refera la cat la suta din material luminat se comporta ca o oglinda. Cu cat obiectul va reflecta mai multa lumina, cu atat mai mult acesta va fi mai stralucitor. Pentru implementarea acestei lumini este nevoie de normal suprafetei obiectului, directia luminii si de pozitia privitorului.

Totodata, implementare Phong realizeaza calculul luminii in fragment shader, astfel am putea spune ca aceasta implementare este o iluminare pe pixeli. Calculul luminii, in acest caz, se realizeaza astfel: se calculeaza in vertex shader normalele varfurilor si se trimit ca si parametrii in fragment shader.

Sursele de lumina pot fi de mai multe feluri: lumina directional, lumina punctiforma si lumina de tip far.

Lumina directional se resfrange aspura intregii scene, iar sursa ei se afla la infinit si are razele paralele. Aceasta lumina se foloseste pentru iluminarea globala a camerei.

Lumina punctiforma emite lumina dintr-o pozitie predefinita in jurul acestei surse. O data cu indepartarea de sursa de lumina punctiforma, intensitatea acesteia se diminueaza. Pentru calculul acestei lumini exista doua formule: una liniara si una cuadratica. In acest proiect am folosit formula cuadratica pentru o calitate mai buna a luminii punctiforme.

In cadrul shader-ului este implementata si functionalitatea de lightning maps, care din pacate nu poate fi vazuta. Aceasta tehnica se foloseste de texturile difuze si speculare pentru a genera diferite grade de reflexive a luminii pentru obiectele cu materiale diferite (de ex. Piatra si sticla).

* Implementarea umbrelor

Cea mai des intalnita tehnica de implementare a umbrelor este “Shadow Mapping”. Aceasta tehnica este una usor de implementat si ofera rezultate relative bune.

Scena trebuie desenata de doua ori, calculand umbrele. In primul rand, se deseneaza scena din punct de vedere al luminii. Aici conteaza doar pozitia obiectelor, nu si culoarea acestora. In acest pas se folosesc valorile de adancime ale varfurilor si anume, cele de pe axa Z. Calculele facute pe aceste coordinate sunt realizate de niste shadere special, care sarveaza volorile pentru urmatoarea etapa de calcul.

Se deseneaza acum scena din punct de vedere al viewer-uli. Aici se compara adancimile fragmentelor cu adancimile salvate in harta de adancime realizata la primul pas. Acele fragmente care au o adancime mai mare in comparative cu cea din harta de adancime nu sunt vizibile din punctul de vedere al privitorului, astfel acestea se afla in umbra.

In acest proiect, mai este prezenta si tehnica “percentage-closer filtering” (PCF) pentru imbunatatirea calitatii umbrelor. Aceasta tehnica presupune esantionarea hartii de adancime de mai multe ori cu cordonate modificate ale texturii. Se verifica pentru fiecare coordonata daca se afla in umbra sau nu, iar la final se face o medie aritmetica a rezultatelor, ootinandu-se astfel umbre mai fine.

* + 1. Motivarea abordarii ales

Implementarea tehnicii “CubeMap” a fost aleasa deoarece creeaza un fundal mai calatativ decat un simplu obiect de tip skybox, generat in 3DS Max. Limitele skybox-ului generat cu tehnica “CubeMap” tind spre infinit, astfel nu se poate iesi “din decor”, lucru ce se intampla atunci cand se foloseste un obiect de tip skybox generat in 3DS.

Am ales modelul Phong pentru iluminare deoarece acesta ofera o calitate mult mai buna a luminii decat cea pe care o reda modelul Gourand, deoarece primul dintre acestea reaizeaza calculele pentru lumina pe pixel si nu pe varfuri in vertex shader. Proiectul are in compozitie o lumina directionala, calculata in fragment shader in functia computeLightComponents si una punctiforma, calculate in functia CalcPointLight. Pentru a se observa mult mai bine lumina punctiforma, aceasta a fost mutate de la o pozitie predefinata pe camera.

Pentru realizarea umbrelor, am optat pentru tehnica PCF, deoarece aceasta taie in proportie marginile drepte ale umbrelor, aducand umbrele intr-o stare mult mai apropiata de cea naturala. Totodata, pentru calitatea umbrelor, am folosit o harta de adancime de 16384 pe 16384, tehnica implementata fiind astfel “Shadow Mapping”.

Pentru a adduce un plus de realism scenei 3D, am adaugat si efectul de ceata, care poate fi oprit sau pornit de pe tastele 4 si respective 5.

* 1. Structuri de date

In realizarea proiectului m-am folosit de reprezentarea datelor oferita de libraria OpenGL. Aceasta contine numeroase functii pentru realizarea operatiilor asupra obiectelor 3D, printer care functiile de translatie, rotatie si scalare.

O alta forma de reprezentare a datelor sunt uniformele folosite in shadere si in aplicatie, pentru transmiterea datelor catre shadere si respective, intre shader.

Totodata am mai folosit o structura de date pentru implementarea luminii punctiforme. Aceasta structura contine campuri pentru pozitia luminii, pentru componentele ambientala difuza si speculara si campuri pentru coeficientii liniari constant si cuadratic.

* 1. Ierarhia de clase

Aplicatia OpenGL este structurata pe sase clase. Acestea sunt: Model3D, OpenGL\_4\_Application\_VS2015, Camera, Skybox, Mesh si Shader.

1. Clasa Model3D

Cu ajutorul acestei clase se ot incarca modele 3D in aplicatia OpenGL. Totodata aeasta clasa incarca si texturile atasate obiectelor, acestea gasindu-se in fisierul “.mtl”. In absenta acestui fisier, sau a texturilor, obiectul va fi incarcat cu o textura neagra. Pentru incarcarea obiectelor, clasa Model3D accepta obiecte in format “.obj”.

1. Clasa OpenGL\_4\_Application\_VS2015

Aceasta reprezinta clasa “main” a aplicatiei. Aici pot fi incarcate obiecte 3D, shadere, aici se pot efectua operatii asupra modelelor 3D si sunt transmise datele catre shadere. Totodata, aici, in cadrul metodei renderScene() se deseneaza scena. Aceasta metoda este apelata in bucla pana la oprirea aplicatie. Apelul in bucla este realizat in functia main(), prezenta si ea in aceasta clasa.

1. Clasa Camera

In aceasta clasa este implementata camera, adica punctul din care se vizualizeaza scena 3D. Aici se calculeaza directia camerei. Aceasta este recalculate o data cu fiecare schimbare a pozitiei mouse-ului sau comutarilor de la tastatura de pe tastele W, A, S si D. In aceasta clasa este definita si pozitia camerei, punctul inspre care este indreptata camera si directia camerei.

1. Clasa Skybox

Aici se deseneaza cubemap-ul care cuprinde scena de obiecte 3D.

1. Clasa Mesh

Clasa ajuta la desenarea obiectelor 3D in scena si la initializare obiectelor de buffer.

1. Clasa Shader

In aceasta clasa se citesc fisierele shader, adica cele cu terminatie “.vert” (vertex shader) si “.frag” (fragment shader). Tot in aceasta clasa sunt compilate shader-ele si raportate eventualele elori de compilare in consola. Folosindu-ne de aceasta clasa, putem incarca mai multe shadere. Incarcarea mai ultor shadere se regaseste la folosirea umbrelor, unde scena trebuie desenata de doua ori folosind doua perechi de shadere.

# Prezentarea interfetei grafice / manual de utilizare

Aaplicatia OpenGL poate fi utilizata cu o serie de comenzi, care vor fi prezentate mai jos:

* Pentru deplasearea camerei: tastele WASD (W - inainte, A - dreapta, S - spate, D - stanga);
* Camera poate fi rotita cu ajutorul mouse-ului;
* Pentru pornirea luminii: tasta 2;
* Pentru oprirea luminii : tasta 1;
* Pentru a porni animatia de tip bechtest: tasta UP;
* Pentru a opri animatia de tip benchtest: tasta DOWN;
* Pentru a porni efectul de ceata se foloseste tasta 5;
* Pentru a opri efectul de ceata se foloseste tasta 4;
* Pentru a comuta intre modurile de vizualizare: solid – 7, wireframe – 8, polygonal – 9;

# Concluzii si dezvoltari ulterioare

In urma implementarii acestei aplicatii, am ajuns sa cunosc o serie de tehnici pentru implementarea unor efecte asupra obiectelor 3D (lumini, umbre, ceata, etc.). Totodata, am dobandit cunostinte referitoare la libraria OpenGL si am invatat sa folosesc unele din functiile oferite de aceasta librarie (translate, rotate, scale, etc.) pentru a realiza operatii cu obiectele 3D. Tot datorita acestei aplicatii, am inteles ce sunt shader-ele si cum functioneaza acestea si am dobandit si cunostinte despre limbajul de programare al shader-elor, GSLS.

Lucrand la aplicatie, cu ajutorul laboratoarelor, am observant diferenta dintre doua modele de iluminare (Phong si Gourand). Totodata, am dobandit cunostinte despre tipurile de lumini existente si cum influenteaza acestea scena.

Tot cu ajutorul laboratorului, am reusit sa implementez tehnica “ShadowMapping”, pentru realizarea umbrelor, tehnica “CubeMap” si efectul de ceata.

Ca si adaos pentru aceste cunostinte, pot spune ca am ajuns sa stapanesc programul de modelare 3DS Max la nivel de incepator. Acest program m-a ajutat enorm in realizarea si modelarea scenei 3D si mi-a crescut apetitul pentru modelarea 3D.

Exista o sumedenie de dezvoltari ulterioare care pot fi aduse aplicatiei, printer care:

* Realizarea citirii a mai multor formate de obiecte;
* Realizarea citirii a mai multor tipuri de texture (de ex. Dds);
* Introducerea in scena a unui numar mai mare de obiecte;
* Generarea unor animatii mai relevante pentru seria Star Wars (sabii laser, batalii cu nave de lupta, arme care sat raga cu laser, etc.);
* Adaugarea mai multor surse de lumina;
* Etc.

# Referite

* http://cgis.utcluj.ro/
* http://tf3dm.com/
* https://learnopengl.com
* http://www.opengl-tutorial.org/