FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

Proiect

la disciplina

**Sisteme de prelucrare grafica**

Titlul

„Baza militara”

Nume : Trif Gheorghe Andrei Indrumator *:*Nandrea Cosmin

Grupa: 30235 Data : 12.01.2020

Cuprins

[1 Prezentarea temei 3](#_Toc29797511)

[2 Scenariul 3](#_Toc29797512)

[2.1 Descrierea scenei si a obiectelor 3](#_Toc29797513)

[2.2 Functionalitati 3](#_Toc29797514)

[3 Detalii de implementare 4](#_Toc29797515)

[3.1 Functii si algoritmi 4](#_Toc29797516)

[3.1.1 Solutii posibile 5](#_Toc29797517)

[3.1.2 Motivarea abordarii alese 6](#_Toc29797518)

[3.2 Modelul grafice 6](#_Toc29797519)

[3.3 Structuri de date 7](#_Toc29797520)

[3.4 Ierarhia de clase 7](#_Toc29797521)

[4 Manual de utilizare 7](#_Toc29797522)

[5 Concluzii si dezvoltarii ulterioare 8](#_Toc29797523)

[6 Referinte 9](#_Toc29797524)

# Prezentarea temei

Tema aleasa pentru acest proiect este crearea unei baze militare.Pentu ca acest proiect consta in creearea si interactionare cu o scena grafica creata in OpenGL , am ales ideea implementari unei baze militare.

# Scenariul

## Descrierea scenei si a obiectelor

In aceasta scena toate obiectele sunt 3D avand textura.Astfel scena contine :doua case ,trei corturi militare , trei tancuri , un jeep, un turn ,gard de proectie si plante.[2].Toate aceste obiecte sunt asezate pe un plan folosit derept pamant care este in interiorul unui skybox.Pentru a putea edita fiecare dintre aceste obiecte am folosit Blender 2.79 .Acesta este o unealta care permite importul ,editarea si exportul diferitelor tipuri de extensii ale obiectelor.[3]

## Functionalitati

Miscarea in scena se face cu ajutorul mouse-ului si a tastaturi , permitand vizualizarea scenei din orice punct.Scena mai poate fi vizualizata in modul wireframe si point folosind tastele R-pentru modul point, T-pentru modul wireframe, Y-pentru normal.In scesa sunt prezente si doua surse de lumina controlate de pe tastele J si L.

O alta functionalitate este posibilitatea vizualizari sub forma unei animatii de prezentarea.Apasand tasta C aceasta va incepe sa ruleze .Aceasta animatie are rol de prezentare parcurcand astfel scena si prezentand toate obiectele.

Nu in ultimul rand in aplicatie este inclusa si ceata ,ceasta nu are o tasta de pornire /oprire avand rolul de a aduce scenei un plus de realism.

# Detalii de implementare

## Functii si algoritmi

OpenGL este considerată o interfață de programare a aplicațiilor (API), utilizată pentru a dezvolta aplicații grafice accesând funcțiile disponibile în hardware-ul grafic. Cu toate acestea, OpenGL este o specificație dezvoltată și întreținută de Grupul Khronos. Specificația OpenGL descrie care este rezultatul dorit sau ieșirea fiecărei funcții. Implementarea reală poate fi diferită între diferitele biblioteci OpenGL. Aceste biblioteci sunt implementate în principal de producătorii de plăci grafice.

Un obiect 3D este reprezentat folosind informații topologice ale vârfurilor. Pentru fiecare vârf avem nevoie de atribute cum ar fi poziția, culoarea, coordonatele de textură și alte informații relevante. Datele de vârf sunt stocate în obiecte Vertex Buffer (VBO).

Pentru a putea rasteriza obiecte 3D, trebuie să scriem cel puțin un vertex shader și fragment shader. Pentru aceasta avem nevoie să:

* generam un obiect shader
* atașăm codul sursă al shader-ului la obiectul de tip shader
* compilăm shader-ul

După ce am compilat programele vertex shader și fragment shader, trebuie să le combinăm într-un așanumit obiect de tip program shader. Pașii pentru aceasta sunt după cum urmează:

* creăm un program shader
* atasăm shader-ele compilate anterior
* operația de linking a shader-elor

Transformările (translație, rotație, scalare) necesare pentru poziționarea modelului într-o scenă se mai numesc și transformări de model. Ele transformă vârfurile obiectului din spațiul local în spațiul global. Transformarea de translație este folosită pentru a muta un obiect (punct) cu un anumit deplasament. Transformarea de scalare mărește sau reduce dimensiunile unui obiect. Transformarea este relativă la origine. Această transformare rotește un obiect cu un unghi dat. Această transformare este, de asemenea, relativă la origine.Cu ajutorul acestor transformari se va putea modifica obiectul in scena dupa adaugare.

Dupa adaugarea unui obiect in scena acesta va avea nevoie de o textura . Texturile sunt în principiu hărți de pixeli 2D încărcate în memoria video, care pot fi apoi asociate unui obiect 3D dat. Asocierea se face prin atribuirea fiecărui vârf 3D unui punct din cadrul imaginii 2D.

Pentru a utiliza o textură, trebuie să o încărcați mai întâi dintr-un fișier extern:

* Se incarca datele de imagine dintr-un fișier imagine (de preferință ".png" sau ".tex" - întotdeauna cu lățime și înălțime egale cu o putere de 2
* Generați un ID de textura
* Activați textura
* Generați o textură din datele imaginii
* Configurați parametrii de textură
* Pentru a aplica textura pe un obiect, este suficient să o legați înainte de a desena obiectul

Dupa adaugarea obiectelor si a texuri este necesar si adaugarea unei surse de lumina .Pentru a putea face acest lucru este necesar sa construi treptat shaderele pentru implementarea modelului de iluminat clasic OpenGL (iluminat Gouraud) folosind limbajul GLSL. Deoarece modelul clasic calculează iluminarea pentru fiecae vârf, shaderul pe care îl vom schimba treptat este Vertex Shader.

Modelul de iluminare Blinn-Phong este o versiune ușor modificată a modelului Phong. Scopul său este de a îmbunătăți reflexiile speculare în anumite condiții, cum ar fi un coeficient foarte scăzut de strălucire (rezultând astfel o zonă speculară mare). Reflexiile speculare ale modelului Phong tind să fie întrerupte imediat ce unghiul dintre direcția de vizionare și reflexia luminii crește peste 90.0 grade.

Urmatorul pas este adaugare de umbre. Există o mulțime de tehnici moderne care pot fi folosite pentru a produce umbre (calculul umbrelor) și toate acestea pot fi implementate în OpenGL. Cu toate acestea, nu există algoritmi perfecți în timp real pentru acest lucru, fiecare având propriile sale avantaje și dezavantaje. Pentru acest laborator, vom explora o tehnică ce oferă rezultate decente și este ușor de implementat, și anume shadow mapping. Shadow mapping este o tehnică multi-trecere care utilizează texturi de adâncime pentru a decide dacă un punct se află în umbră sau nu. Cheia este aceea de a observa scena din punctul de vedere al sursei de lumină în loc de locația finală de vizionare (locația camerei). Orice parte a scenei care nu este direct observabilă din perspectiva luminii va fi în umbră.

Etapele principale ale algoritmului sunt descrise mai jos:

1. Rasterizarea scenei din punctul de vedere al luminii. Nu contează cum arată scena (informații despre culoare); singurele informații relevante în acest moment sunt valorile de adâncime. Aceste valori sunt stocate într-o hartă de umbră (sau hartă de adâncime) și pot fi obținute prin crearea unei texturi de adâncime, atașarea la un obiect framebuffer și rasterizarea întregii scene (așa cum este văzută din poziția luminii) în acest obiect. În acest fel, textura de adâncime este umplută direct cu valorile relevante ale adâncimii.
2. Rasterizarea scenei din punct de vedere al observatorului (poziția camerei). Se compară adâncimea fiecărui fragment vizibil (proiectat în cadrul de referință al luminii) cu valorile de adâncime din harta umbrelor. Fragmentele care au o adâncime mai mare decât cea care a fost stocată anterior în harta de adâncime nu sunt direct vizibile din punctul de vedere al luminii și sunt, prin urmare, în umbră

Acum pentru a avea o scena completa mai trebuie sa adaugam un fundal care reprezinta partea scenei pe care nu am modelat-o. Putem realiza acest lucru prin includerea scenei noastre 3D în interiorul unui cub cu texturi diferite aplicate pe fiecare față. Acest cub se numeste skybox și este compus din 6 texturi.[1].

### Solutii posibile

Dupa cum am mentionat la inceput aceasta este o aplicatie in care se foloseste OpenGl [4].Pentru a implementa o astfel de scena se puteau folosi diferite tool-uri si limbaje precum Unity.Eu am ales aceasta abordare deoarece notiunile care m-au ajutat sa o finalizez sunt prezentate in laborator.Astfel am inceput prin cautarea obiectelor 3D specifice temei mele .Odata gasite aceste obiecte au fost introduse in blender 2.79 cu ajutorul caruia am definit scena si am editat cateva obiecte.Dupa aceasta am construit scena in blender am exportat fiecare obiect cu extensia .obj.Astfel sa creat folderul obiectului care continea texturile fisierul .obj si fisierul .mtl. In aplicatie am introdus posibilitatea de a te misca cu mousul si tastatura si modul de vizualizare .Odata finalizate aceste lucruri am adaugat obiectele si texturile .

In aceasta faza scena era in mare parte terminata singurul lucru fii adaugarea efectelor,a luminii si a umbrelor care au fost facute conform indicatiilor din laborator.

### Motivarea abordarii alese

Am ales abordarea descrisa mai sus deoarece informatiile necesare au provenit din notiunile prezentate la laboratorul de prelucrare grafica .Astfel fiind foarte simpu sa aprofundez aceste notiuni .Suportul de lucru al acestui laborator este foarte bun si usor de inteles pentru oricine incearca sa inteleaga OpenGl si sa creeze o aplicatie grafica.

## Modelul grafice

Modelul grafic al acestei scene a fost creat folosit obiecte 3D descarcate de pe internet si modificate in blender 2.79 ,dupa aceea au fost adaugate in scena conform notiunilor invatate la laborator .



## Structuri de date

OpenGL Mathematics (GLM) este o bibliotecă de matematică C++ (compusă doar din fișiere antet – „header”) pentru software-ul grafic bazat pe specificațiile OpenGL Shading Language (GLSL). De altfel, această bibliotecă este utilă în orice context de dezvoltare software care necesită operații matematice simplu de utilizat.

Structurile de date folosite sunt prezentate si in laborator, dar cele mai folosite sunt ex glm::vec4 newVector(1.0f, 2.0f, 3.0f, 1.0f); care creeaza un vector coloana cu 4 elemente:x y z w.Astfel pentru a declara vectori cu 2 sau 3 elemente se vor folosi vec3 ,vec2.

Matricile acestea pot fi definite într-un mod similar folosind celelalte tipuri de date tip matrice pe care le oferă GLM. Matricele pătratice pot fi de asemenea definite folosind tipurile de date mat2, mat3 și mat4. Acestea pot fi inițializate cu aceeași valoare pe diagonală folosind constructorii cu un singur parametru. De exemplu, o matrice identitate cu dimensiunea 4 x 4 poate fi declarată ca: glm::mat4 identity4Matrix(1.0f);

## Ierarhia de clase

Clasele acestei aplicatii sunt:

* Camera -clasa folosita pentru determinarea matricei de view;
* Model3D-deseneaza fiecare mesh din model;
* Mesh - folosit la setarea texturilor si la initializarea buffar-objects;
* Shader-folosit la procesarea tuturor fisierelor shader;
* SkyBox - folosit la desenarea parti nemodelate din scena;

# Manual de utilizare

Aceasta aplicatie este scrisa in Visual Studio 2017 .Pentru a putea rula aceasta aplicatie codul c++ trebuie compilat si rulat.O data rulat se va deschidea ferasta de vizualizare.In aceasta fereasta se va putea modificata directia de vizualizare cu ajutorul tastelor W ,A, S, D si cu ajutorul mouse-ului.Cu ajutorul tastelor R,T,Y se va putea modivica modul de vizualizare .Astfel tasta R -este pentru modul point,T-moul wireframe,Y-modul normal.Cu ajutorul celor tastelor J,L si N ,M se vor putea roti cele doua surse de lumina din iteriorul scenei,iar cu ajutorul tastei C se va activa animatia de prezentare.

# Concluzii si dezvoltarii ulterioare

Acest proiect a avut un impact benefic asupra mea doearece am reusit sa inteleg obiectivele pe care trebuie sa le ating pentru a putea implementa o scena grafica.A fost foarte interesat sa descoper cat de uimitore este munca unui artst 3D si cat de mult te poti distra cu ajutorul unei scene si putina imaginatie.

O idee de dezvoltare ulterioara a acestei aplicatii ar fii adaugarea unor obiecte cu rol specific pentru mediul inconjurator : mai multe plante,drumuri etc.Cat si adaugarea unor efecte precum vant sau ploaie care ar adauga un plus de realism scenei.

O alta idee de dezvoltare interesanta ar putea fi adaugarea personaj precum in jocurile FPS care sa poata face anumite actiuni.

# Referinte

[1]. Indrumator de laborator-https://moodle.cs.utcluj.ro/course/view.php?id=186 ;

[2].Sit-uri de obiecte 3D -https://free3d.com , https://www.cgtrader.com

[3]. Tutorial blender- https://docs.google.com/document/d/1njtWPMmOQNIaD\_z9ve8iPRUqQTWdIV\_PO-NvPD0nOuM/edit#heading=h.7h28ckb0ku81

[4].Tutorial opengl - http://www.opengl-tutorial.org