**Proiect prelucrare grafica**

Student:Tulbure Claudiu

Grupa:30234

Contents

[1.Prezentarea temei 2](#_Toc124724067)

[2.Scenariul 3](#_Toc124724068)

[3.Detalii de implementare 9](#_Toc124724069)

[3.1Functii si algoritmi 9](#_Toc124724070)

[3.2Modelul Grafic 10](#_Toc124724071)

[3.3 Structuri de date 11](#_Toc124724072)

[3.4 Ierarhie de clase 11](#_Toc124724073)

[4.Manual de utilizare 12](#_Toc124724074)

[5.Concluzii si dezvoltari ulterioare 12](#_Toc124724075)

[6.Referinte 13](#_Toc124724076)

# 1.Prezentarea temei

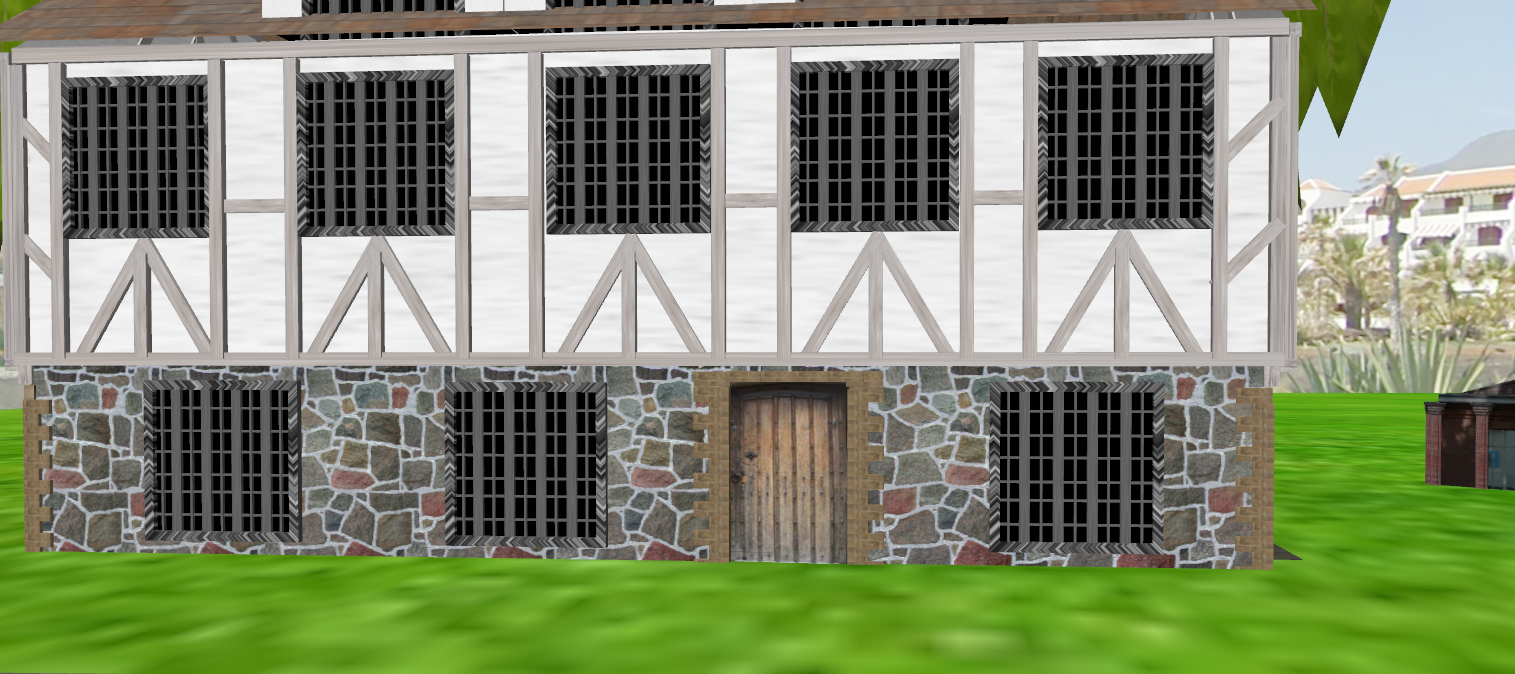
Proiectul are ca si scop realizarea unei scene 3D cu ajutorul bibliotecii OpenGL,a librariilor GLM , GLFW , si Blenderul pentru creearea scenei.In crearea acestei aplicatii ,am implementat 2 surse de lumina,3 moduril de vizualizare (solid,point,wireframe),maparea texturilor ,animatii ,si efecte de ceata si ploaie.

# 2.Scenariul

Descrierea scenei si a obiectelor:



In imaginea de mai sus putem observa o imagine de ansamblu cu intreaga scena.Obiectele statice din scena,cele asupra carora nu se pot efectua miscari sunt:statuia cu leu,copacii,garajul ,casele din mijlocul scenei,terenul de fotbal,si monstrul de langa terenul de fotbal.Toate aceste obiecte sunt de design ,si nu suporta functionalitati.Ideea a fost de a creea o localitate cu case,garaj ,parc si teren de fotbal.



In imaginea de mai sus se poate observa o casa medievala.

Aceasta casa este mapata intr-un singur obiect cu intreaga scena,insa usa nu.Usa este un obiect exportat separate din blender,ca sa poate suporta functionalitatea de a se deschide dupa o serie de rotatii si translatii,pentru a se putea intra in casa.



In aceasta imagine putem observa o eoliana,la fel ca si casa,eoliana este mapata la obiectul scenei,mai putin eliceele sale,care sunt mapate intr-un obiect separate,ca sa se poata roti la apasarea unui buton.



In aceasta imagine putem observa o lampa,este exportata separate din blender,si este folosita ca centru al luminii punctiforme.



In aceasta imagine putem observa un nor ,este un obiect exportat separate,alaturi de picaturile de ploaie.Functionalitatile indeplinite sunt de a creea efectul de ploaie,prin translatarea picaturilor de ploaie in jos.



Mai sus se poate observa obiectul care joaca rolul de soare.Este mapat separate din blender,iar functionalitatea sa este de a fi centrul luminii directionale,odata cu miscarea sa se va misca si lumina directionala.



In ultima imagine se poate observa usa de la garaj.

Functionalitatea indeplinita de aceasta este de a se deschide sau inchide la apasarea unui buton.

# 3.Detalii de implementare

# 3.1Functii si algoritmi

Pentru animarea anumitor obiecte descarcate de pe internet(de exemplu usile care se pot deschide ,elicea care se invarte),a fost nevoie sa le decupez in Blender si sa le export separat ,fiecare obiect de acest tip avand functia specific de randare si incarcare.Functiile care primesc ca si argument shaderul se ocupa de desenarea obiectului respectiv.De asemenea ,pentru a face ca un obiect sa se miste in jurul anumitor axe raportat la originea sa,a fost nevoie mai intai sa il translatez in origine,sa realizez rotatia/translatia/scalarea si apoi o translatie inversa.Pentru controlul anumitor obiecte,s-au folosit niste flaguri cu ajutorul functiei processMovements,si in functia de aceste flaguri s-a randat scena.

Pentru realizarea animatiei de prezentare,am folosit un flag care daca era activat afisam toate actiunile facute intr-un fisier,de exemplu am scris in fisier cifra 1cand ma misc la stanga cu camera.Dupa ce am terminat de parcurs mapa,am oprit acel flag si am setat fisierul ca fisier de intrare ,astfel incat atunci cand am setat flagul de animatie al prezentarii,sa citesc actiunile facute .De asemenea ,fiecare animatie va porni din aceeasi pozitie.In timpul animatiei,toate actiunile vor fi oprite.

Pentru implementarea luminii,am folosit algoritmul lui Phong studiat la laborato.Acest algoritm presupune ca lumina e alcatuita din 3 componente : lumina ambientala (valoarea sa poate fi precalculata ca un effect global,iar calculul sau nu depinde de pozitia luminii sau a spectatorului),lumina difuza(calculul sau depinde de normala suprafetei si de directia sursei de lumina,dar nu de pozitia spectatorului) si lumina speculara (aceasta componenta da efectul de stralucire ,iar calculul sau depinde inclusive de directia de vizualizare).

In proiect am implementat 2 tipuri de lumina ,si anume lumina directionala(care e situata la infinit ,iar razele sale sunt paralele si au aceeasi directie) si lumina punctiforma(care are o anumita pozitie in spatiu,ilumineaza radial si este estompata in functia de distanta).

Pentru realizarea efectului de ceata,am folosit laboratorul special dedicate acesteia,pentru ploaie am introdus un nor ,si mai multe obiecte de tip ploaie,care se translateaza in continuu.De asemenea,ceata folosita a fost cea exponential patratica :

(fog = exp(-pow(fragmentDistance\*fogDensity))),calculate si returnata de functia computeFog din shaderul basic,ca mai apo isa apelam functia de mix care va interpola culoarea cu valoarea cetei exponential patratice.

Pentru Camera.cpp ,implementarea arata asftel:pentru a inainta ,formula pentru directie = cameraFrontDirection\*(-speed),pentru miscarea inapoi dir = cameraFrontDirection\*speed , pentru miscarea la dreapta dir=cameraRightDirection\*speed , pentru miscarea la stanga dir = cameraRightDirection\*(-speed),pentru miscarea in sus dir = cameraUpDirection\*speed,pentru miscarea in jos dir = cameraUpDirection\*(-speed).La final,adunam dir la cameraPosition si cameraTarget.

Pentru rotatia camerei ,am folosit o serie de translatari si rotatii.Am definit o matrice de translatie cu cameraPosition,apoi o rotatie in sus cu unghiul yaw,apoi o rotatie cu unghiul pitch folosind coordonatele cameraRightDirection.Am aplicat aceasta transformare ,si am atribuit valoarea rezultata la cameraTarget,si am actualizat noile coordinate ale vectorilor de cameraFrontDirection si cameraRightDirection.

# 3.2Modelul Grafic

Diagram, PowerPoint

Description automatically generated

In imaginea de mai sus putem observa Pipeline-ul programabil din OpenGL.Acesta conține cel puțin stagiile de vertex shading și fragment shading, și înlocuiește pipeline-ul fix, care nu mai este suportat (iluminare și transformări). Faza de vertex shading procesează fiecare vârf independent de celelalte; datele vârfurilor sunt primite de la aplicație prin obiecte de tip vertex buffer (VBO). Faza de rasterizare generează un set de fragmente care sunt procesate în faza de fragment shading; ieșirea este reprezentată de culoarea și adâncimea fragmentului (coordonata Z). Unele etape (cum ar fi primitive setup, clipping și rasterization) nu sunt programabile. Tesselation Shader și Geometry Shader sunt opționale.

# 3.3 Structuri de date

Am folosit diverse flaguri,care sa ajute la reazlizarea unei actiuni la apasarea unui anumit buton.Pentru fiecare obiect am definit o noua matrice de model si cate un obiect de tip Model3D.Pentru picaturile de ploaie ,mi-am definit un vector de obiecte de tip ploaie,cate un model pentru fiecare picture.Shaderele folosite au fost cele oferite de template modificate astfel incat sa functioneze luminile si efectul de ceata,precum si shadere noi pentru SkyBox.

Am folosit matrici ,care reprezentau modelul fiecarui obiect in parte(glm::mat4).Vectori pentru stocarea coordonatelor si a culorilor(glm::vec3),doua obiecte de tip Camera,unul fiind folosit pentru a fi camera fixa la animatie.In BasicShader am folosit o structura de tip PointingLight pentru stocarea caracteristicilor luminii punctiforme.

# 3.4 Ierarhie de clase

Clasele folosite au fost doar cele oferite de template-ul de pe moodle.Clasa de Camera este clasa care se ocupa de miscarile camerei.

Clasa Mesh este clasa responsabila de incarcarea texturilor.Clasa Model3D se ocupa de incarcarea unui nou model3D (a unui obiect ) din blender.Clasa Shader este clasa destinate special shaderelor,si se ocupa de incarcarea si compilarea lor.Clasa SkyBox este responsabila de incarcarea “cubului” in care se afla scena.Clasa Window este responsabila de creearea ferestrei de vizualizare.

# 4.Manual de utilizare

1. Se va rula executabilul
2. Va aparea mapa,cu camera pozitionata intr-un anumit punct
3. Tastele speciale sunt:
   1. UP\_KEY = se misca in fata
   2. DOWN\_KEY = se misca in spate
   3. LEFT\_KEY = se misca la stanga
   4. RIGHT\_KEY = se misca la dreapta
   5. W = se roteste camera in sus
   6. S = se roteste camera in jos
   7. A = se roteste camera la stanga
   8. D = se roteste camera la dreapta
   9. GLFW\_KEY\_9 = activeaza animatia de prezentare
   10. GLFW\_KEY\_P = activeaza ploaia
   11. GLFW\_KEY\_O = dezactiveaza ploaia
   12. GLFW\_KEY\_8 = mod de vizualizare al luminii punctiforme
   13. GLFW\_KEY\_7 = mod de vizualizare al luminii directionale
   14. GLFW\_KEY\_N = creste densitatea cetei
   15. GLFW\_KEY\_M = scade densitatea cetei
   16. GLFW\_KEY\_1 = deschide sau inchide usa de la garaj
   17. GLFW\_KEY\_2 = deschide sau inchide usa de la casa
   18. GLFW\_KEY\_3 = porneste sau opreste elicea
   19. GLFW\_KEY\_Z = modul de vizualizare va fi point
   20. GLFW\_KEY\_X = mod de vizualizare line
   21. GLFW\_KEY\_C = mod de vizualizare fill
   22. GLFW\_KEY\_J = misca lumina directionala la stanga
   23. GLFW\_KEY\_L misca lumina directionala la dreapta

# 5.Concluzii si dezvoltari ulterioare

Realizarea unui joc 3D nu este simplu,cum m-as fi asteptat,ci destul de dificil,iar pe partea de dezvoltari ulterioare,as dezvolta mult mai mult designul,precum si functionalitatile:as introduce mai multe miscari in scena,efecte mult mai frumoase .

# 6.Referinte

-Lucrari de laborator

- <https://www.opengl.org/>