**O imagine care conține text, Font, siglă, Grafică

Descriere generată automat**

PRELUCRARE GRAFICA \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NISTOR DALIA EMILIA

GRUPA 30234

Cuprins

[1.Prezentarea temei 3](#_Toc156154834)

[2.Scenariul 3](#_Toc156154835)

[2.1 Descrierea scenei si a obiectelor 3](#_Toc156154836)

[2.2 Functionalitati 4](#_Toc156154837)

[3. Detalii de implementare 4](#_Toc156154838)

[3.1 Functii si algoritmi 4](#_Toc156154839)

[3.1.1 Solutii posibile 5](#_Toc156154840)

[3.1.2 Motivarea abordarii alese 5](#_Toc156154841)

[3.2 Modelul grafic 5](#_Toc156154842)

[3.3 Structuri de date 5](#_Toc156154843)

[3.4 Ierarhia de clase 5](#_Toc156154844)

[5. Prezentarea interfeței grafice utilizator / manual de utilizare 6](#_Toc156154845)

[6. Concluzii și dezvoltări ulterioare 6](#_Toc156154846)

[7.Referinte 6](#_Toc156154847)

# 1.Prezentarea temei

Acest proiect își propune să creeze o reprezentare realistă a obiectelor 3D folosind biblioteca OpenGL, o API (Interfață de Programare a Aplicațiilor) utilizată pe scară largă, cross-platform, pentru programarea graficelor 2D și 3D în programele de calculator. Am ales sa implementez o plaja, cu o multime de palmieri, umbrele si diferite animale.

# 2.Scenariul

## 2.1 Descrierea scenei si a obiectelor

Scena mea este relativ simpla, pe o buna parte din suprafata scenei se afla nisipul iar pe cealalta marea. Am pus mai multi palmieri pentru a da impresia unei zone calde si de asemenea diferite animale cum ar fi testoase, crabi si cativa lei de mare. Pe o parte se afla o zona de plaja cu umbrele si mingi de volei, iar pe cealalta o alta plaja asemanatoare. In plus am adaugat si o barca pentru a face scena mai interactiva.

O imagine care conține în aer liber, plajă, cer, nisip

Descriere generată automat

O imagine care conține în aer liber, umbrelă, cer, plajă

Descriere generată automat

O imagine care conține captură de ecran, cer, Joc de PC, Software de jocuri video

Descriere generată automat

## 2.2 Functionalitati

Utilizatorul poate vedea o previzualizare completă a scenei înainte de a începe să se plimbe liber pe harta. El poate vizualiza întreaga scenă apăsând tastele de pe tastatură și folosind mouse-ul pentru a direcționa camera în poziția dorită. Obiectul barca plasat pe mare poate sa mearga inainte si inapoi, iar de asemenea 2 dintre mingile de volei si unul dintre leii de mare pot fi translatate. In plus, la apasarea unei taste cerul se schimba in modul noapte iar utilizatorul poate sa aiba o vizualizare solid, wireframe si poligonala apasand diferite taste.

# 3. Detalii de implementare

## 3.1 Functii si algoritmi

Biblioteca OpenGL constă într-o varietate mare de funcții, cum ar fi cele preimplementate în nucleul proiectului nostru. Câteva dintre funcțiile importante utilizate în acest proiect sunt:

initOpenGLState(): pentru a activa anumite funcții specifice OpenGL.

initModels(): unde toate modelele sunt încărcate din fișierele corespunzătoare.

initShaders(): unde toate shader-urile sunt încărcate.

initUniforms(): funcția în care legăm matricele model/view/projecție/normal și alte componente cu shader-urile.

initFBO(): pentru calcularea depthMap-ului pentru efectul de umbrire. setWindowCallbacks(): activează funcțiile de apel pentru fereastra, tastatură și mouse. processMovement(): procesează acțiunile de la taste. renderScene(): pentru a renderiza obiectele cu shader-urile corespunzătoare (în interiorul acestei funcții avem apeluri la funcția renderObjects(gps::Shader shader, bool depthPass), care renderizează obiectele noastre conform parametrilor).

### 3.1.1 Solutii posibile

1.Optimizarea Performanței:

Batch Rendering: Implementarea unui sistem de randare în lot (batch rendering) pentru a minimiza numărul de apeluri la funcții de randare și a îmbunătăți performanța.

Culling: Utilizarea tehnicilor de culling (frustum culling, back-face culling) pentru a evita randarea obiectelor care nu sunt vizibile, economisind resurse de procesare grafică.

2.Iluminare și Shading:

Model de Iluminare: Alegerea unui model de iluminare adecvat (de exemplu, Phong) pentru a obține efecte vizuale realiste.

Shader-uri: Utilizarea shader-urilor pentru implementarea unui sistem de iluminare și shading avansat.

3. Gestionarea Memoriei:

Eliberarea Resurselor: Implementarea unui sistem de eliberare a resurselor pentru a preveni scurgerile de memorie.

Optimizarea Texturilor: Utilizarea texturilor în mod eficient pentru a minimiza consumul de memorie video.

### 3.1.2 Motivarea abordarii alese

În laboratoarele a fost o focalizare specifică pe acest subiect, am învățat cum să calculăm iluminarea și umbrele, precum și cum să translăm, scalăm și rotim diverse obiecte. Cu această bază și un contur de proiect clar, sarcinile necesare pentru finalizarea proiectului au fost bine definite.

## 3.2 Modelul grafic

Obiectele și texturile au fost descărcate de pe internet. Majoritatea obiectelor au fost importate în Blender pentru a fi editate și plasate în scena finală.

## 3.3 Structuri de date

Structurile de date utilizate în acest proiect sunt cele comune în limbajele de programare C/C++ și structurile de date din biblioteca glm (cum ar fi vectori, matrice etc.).

## 3.4 Ierarhia de clase

Camera: conține implementarea pentru mișcarea camerei (sus, jos, în față, înapoi, la stânga, la dreapta, rotire stanga si rotire dreapta

Mesh: reprezintă un obiect 3D; include următoarele structuri de date

• Model3D: conține metode pentru afișarea meshe-urilor folosind un program shader specificat

• Shader: conține metode pentru crearea și activarea programelor shader

# 5. Prezentarea interfeței grafice utilizator / manual de utilizare

* W,A,S,D miscarea camerei inainte, inapoi, stanga, dreapta
* U,I miscarea camerei sus jos
* Q,E rotire scena stanga si dreapta
* G,H,J vizualizare scena mod solid, wireframe, si poligonal
* R,B rotire barca dreapta, stanga
* K,L translatie mingi volei sus jos
* N schimbare mod noapte
* T unul dintre leii de mare se deplaseaza in fata

# 6. Concluzii și dezvoltări ulterioare

Proiectul a reușit să atingă obiectivele stabilite și să ofere o implementare funcțională a unui sistem grafic interactiv. Principalele concluzii includ:

Implementare Corectă: S-au implementat cu succes funcționalități precum iluminare, shading, mișcarea camerei și încărcarea eficientă a modelelor.

Structură Modulară: Ierarhia de clase și utilizarea structurilor de date au contribuit la o structură modulară și ușor de extins în viitor.

Dezvoltări Ulterioare: Pentru a extinde proiectul și a îmbunătăți experiența utilizatorului, următoarele direcții pot fi explorate:

Efecte Vizuale Avansate: Adăugarea de efecte vizuale avansate, cum ar fi reflexii, umbre dinamice sau particule, pentru a îmbogăți aspectul general al scenei.

Interacțiune Utilizator Extinsă: Dezvoltarea interacțiunii utilizatorului prin adăugarea de comenzi suplimentare sau interfețe pentru a oferi o experiență mai complexă.

Optimizări de Performanță: Identificarea și implementarea unor optimizări suplimentare pentru a asigura o randare fluidă și eficientă pe o varietate de platforme.

Extinderea Bibliotecii de Obiecte și Texturi: Adăugarea de noi obiecte și texturi pentru a diversifica scena și a oferi mai multe opțiuni pentru utilizatori.

# 7.Referinte

* <https://free3d.com/>
* Indrumatorul de laborator
* <https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM>
* <https://learnopengl.com/>