A picture containing logo

Description automatically generated**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA**

**INFORMATIEI**

**Documentatie Proiect**

**Prelucrare Grafica**

**Student: Pinciuc Darius-Eduard**

**Data: 17.01.2024**

**Grupa: 30236**

CUPRINS

[1. PREZENTAREA TEMEI 3](#_Toc92917554)

[2. SCENARIUL 3](#_Toc92917555)

[Descrierea scenei si a obiectelor 3](#_Toc92917556)

[Functionalitati 4](#_Toc92917557)

[3. DETALII DE IMPLEMENTARE 5](#_Toc92917558)

[Functii si algoritmi 5](#_Toc92917559)

[Modelul grafic 6](#_Toc92917560)

[Structuri de date 6](#_Toc92917561)

[Ierarhii de clase 7](#_Toc92917562)

[4. PREZENTAREA INTERFETEI GRAFICE 7](#_Toc92917563)

[5. MANUAL DE UTILIZARE 8](#_Toc92917564)

[6. CONCLUZII 9](#_Toc92917565)

[7. REFERINTE 9](#_Toc92917566)

# PREZENTAREA TEMEI

**Tema Proiectului**: Dezvoltare 3D OpenGL

Proiectul propus constă în crearea unei scene interactive în mediul OpenGL, oferind utilizatorilor posibilitatea de a explora un mediu tridimensional detaliat. Scena va include diverse elemente, cum ar fi mașini, clădiri, personaje și efecte speciale. Prin intermediul acestui proiect, ne propunem să înțelegem modul în care un calculator procesează o scenă de obiecte și să acumulăm experiență în utilizarea bibliotecilor și librăriilor 3D, precum OpenGL, GLFW și GLM.

Obiective :

1. Crearea scenei 3D
2. Implementarea unei camere
3. Adăugare diverse obiecte(mașini, personaj)
4. Diverse efecte vizuale, precum umbre, ceață și lumini

Tehnologii Utilizate :

* OpenGL, C++, diverse modele de iluminare

# SCENARIUL

## Descrierea scenei si a obiectelor

Scena a pornit de la un gând simplu, tatăl meu și o mică pasiune, mașinile vechi(de epocă). Am modelat și texturat diferite mașini în Blender, iar cu ajutorul tehnologiei implementate de Apple (iPhone 14 Pro Max) l-am scanat pe tatăl meu, apoi l-am exportat ca .obj. Am încercat să surprind realitatea unui mic orășel de provincie cu o mică parcare. Astfel, se pot observa atât obiecte serios texturate , dar și obiecte care să surprindă satira asupra anumitor aspecte din orășelul de provincie (Dacia cu cartoane de la banane în lunetă).

**Functionalitati**

Au fost implementate pentru scenă diferite moduri de vizualizare(ex. Wireframe) și diferite efecte(ex. ceață, diferite lumini). În plus, o mică animație care permite deschiderea ușii unei mașini.



# DETALII DE IMPLEMENTARE

## Ierarhii de clase

* Camera.cpp și Camera.hpp:

Clasa Camera se ocupă de gestionarea perspectivelor și poziționării camerei în spațiu. Aceasta furnizează funcționalități pentru controlul și mișcarea camerei în scenă.

* Mesh.cpp și Mesh.hpp:

Clasa Mesh constituie o structură de triunghiuri care definește geometria unui obiect tridimensional. Această clasă permite stocarea și manipularea vertecșilor, normalelor și coordonatelor texturilor asociate obiectului respectiv.

* Model3D.cpp și Model3D.hpp:

Clasa Model3D administrează un obiect tridimensional compus din una sau mai multe plase (Mesh). Această clasă oferă posibilitatea de a adăuga funcționalități pentru Shader.cpp și Shader.hpp:

**Functii si algoritmi**

* Am utilizat diferite functii existente precum glm::Translate, glm::rotate
* Am utilizat diferiti algoritmi, de exemplu, am utilizat algoritmul Blinn-Phong

**Ceața:**

1. vec4 fPosEye = view \* model \* vec4(fPosition, 1.0f)

- Se transformă poziția fragmentului (“fPosition”) din spațiul local al obiectului (“model”) în spațiul camerei (“view”). Rezultatul este stocat într-un vector 4D (“vec4”) pentru a permite utilizarea transformărilor omogene.

2. float fogDensity = 0.05f;

- Se definește densitatea ceții;

3. float fragmentDistance = length(fPosEye);

- Se calculează distanța a fragmentului față de camera observatorului. Acest lucru se face prin calcularea lungimii vectorului “fPosEye”.

4. float fogFactor = exp(-pow(fragmentDistance \* fogDensity, 2));

- Se calculează factorul de ceață utilizând o funcție exponențială.

5. return clamp(fogFactor, 0.0f, 1.0f);

- Se returnează factorul de ceață, dar acesta este mărginit între 0 și 1 pentru a se asigura că este în intervalul valid.

**Lumini Punctiforme:**

float ambientStrength1 = 0.0001f;, float specularStrength1 = 0.0001f;, float shininess1 = 1.0f;:

Acestea sunt parametrii care controlează intensitatea iluminării ambiante, iluminării specular și coeficientul de strălucire (shininess) al suprafeței.

.

1. vec3 cameraPosEye = vec3(0.0f);:

Se definește poziția camerei în spațiul camerei.

2. vec3 normalEye = normalize(fNormal);

Se calculează normala la suprafața fragmentului și este normalizata;

3. vec3 lightDirN = normalize(lightPosEye.xyz - fPosEye.xyz);:

Se calculează direcția de la fragment la sursa de lumină și este normalizată;

4. float dist = length(lightPosEye.xyz - fPosEye.xyz);:

Se calculează distanța de la fragment la sursa de lumina: float a = 3.0;, float b = 4.0;, 5. float att = 1.0f / (a \* dist \* dist + b \* dist + 1.0f);:

Se calculează atenuarea luminii în funcție de distanța de la sursa de lumină la fragment. Aici, s-a folosit o formă specifică de atenuare quadratică.

6. ambient = att \* ambientStrength1 \* lightColor;

diffuse = att \* max(dot(normalEye, lightDirN), 0.0f) \* lightColor;

specular = att \* specularStrength1 \* specCoeff \* lightColor;:

Se calculează componente ale iluminării: ambientă, difuză și speculară. Aceste componente sunt afectate de atenuare .

7. vec3 viewDirN = normalize(cameraPosEye - fPosEye.xyz);

vec3 halfVector = normalize(lightDirN + viewDirN);

float specCoeff = pow(max(dot(normalEye, halfVector), 0.0f), shininess1);

Se calculează direcția privirii și vectorul jumătate (half-vector) utilizat în iluminarea speculară. Coeficientul specular este calculat folosind strălucirea (shininess) și produsul scalar dintre normală și half-vector.

8. return (ambient + diffuse + specular) \* att \* vec3(2.0f, 2.0f, 2.0f);

Se returnează suma iluminării ambientale, difuze și specular cu atenuarea aplicată. Înmulțirea cu un vector vec3(2.0f, 2.0f, 2.0f) pare să mărească global intensitatea iluminării.

**Structuri de date**

* Am utilizat diferite structuri de date pentru a comunica cu shaderele prin uniform, de exemplu: GLuint, GLfloat.
* Am utilizat si diferite structuri: Model, Shaders pentru a manipula obiecte 3D

**Modelul grafic**

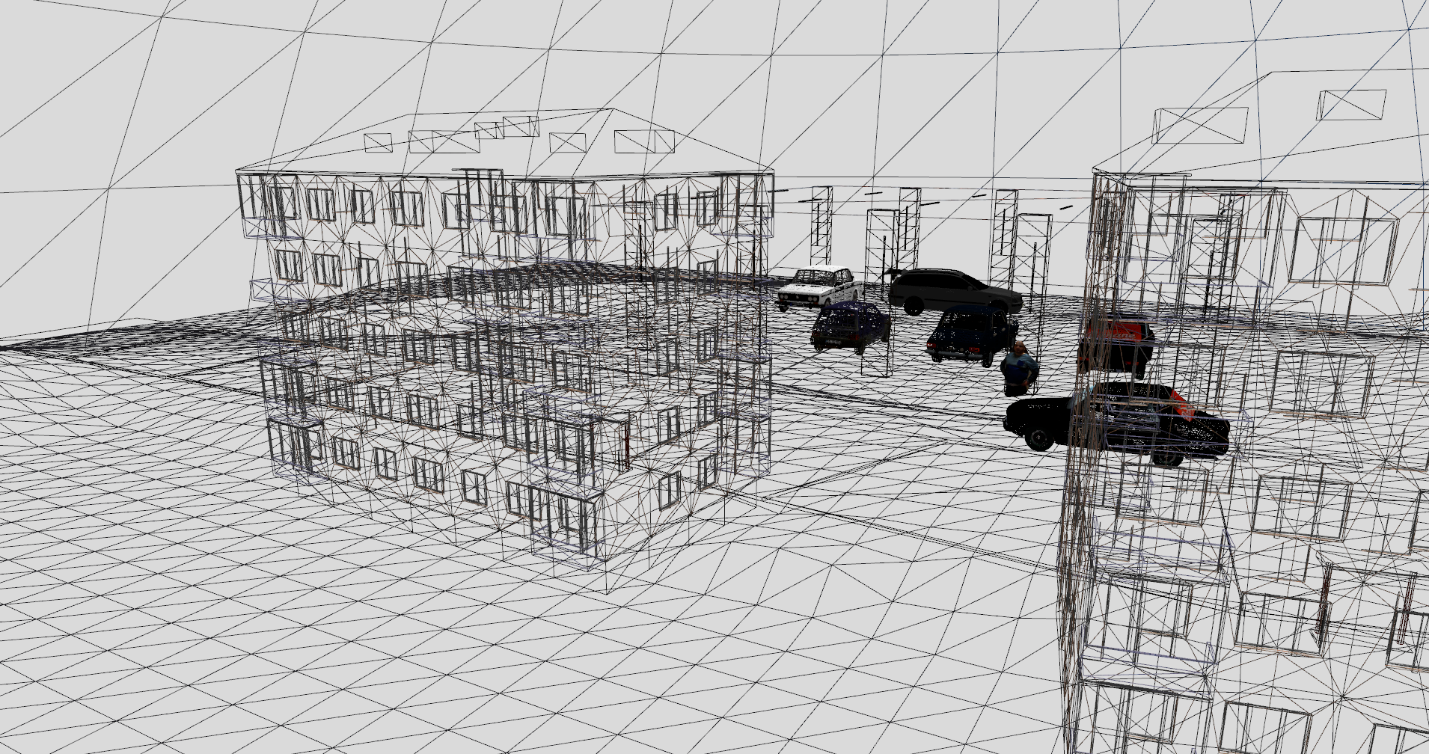
* Am utilizat Blender pentru crearea diferitelor obiecte

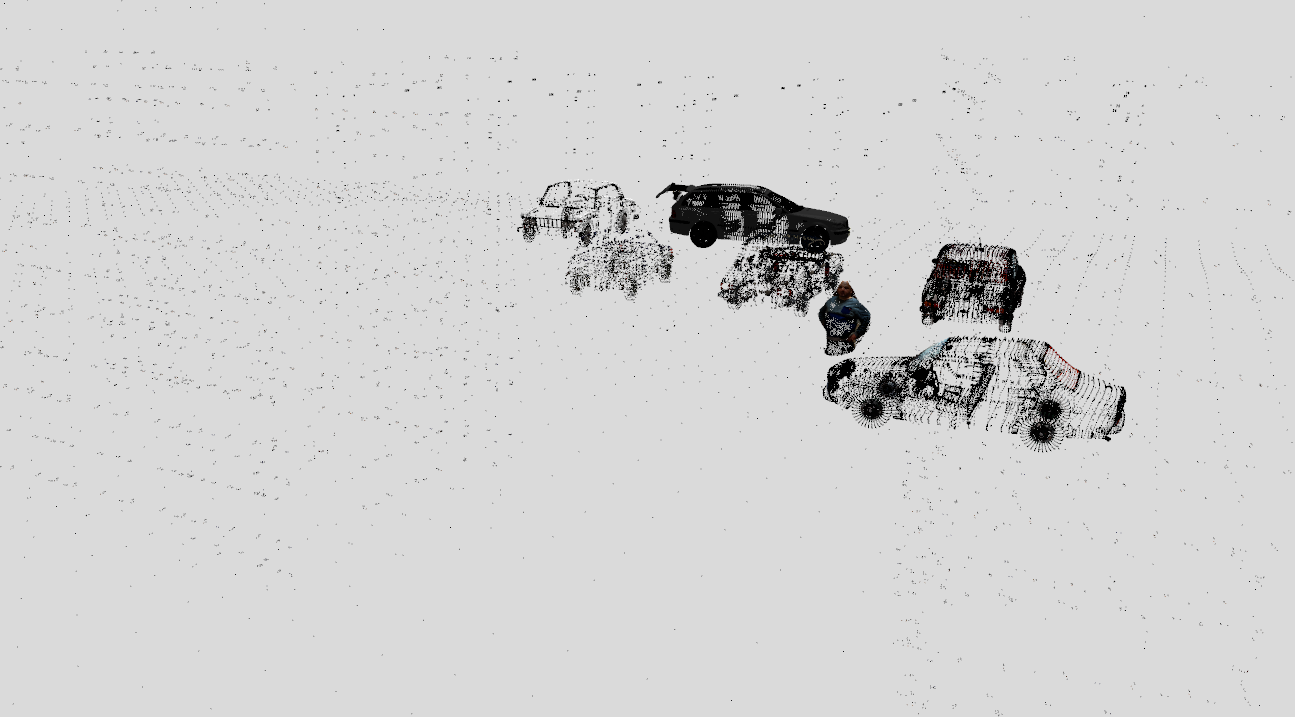
Am modelat scena cu diferite moduri: Sculpt Mode, Edit Mode(Subdivide). In plus, am utilizat UV Editor pentru texturarea destul de complexa a diferitelor obiecte.

* Am creat si un skydome prin crearea si taierea a 2/3 dintr-o sfera, apoi texturarea acesteia cu o imagine specifica(cerul)

# PREZENTAREA INTERFETEI GRAFICE

**Moduri de vizualizare:**

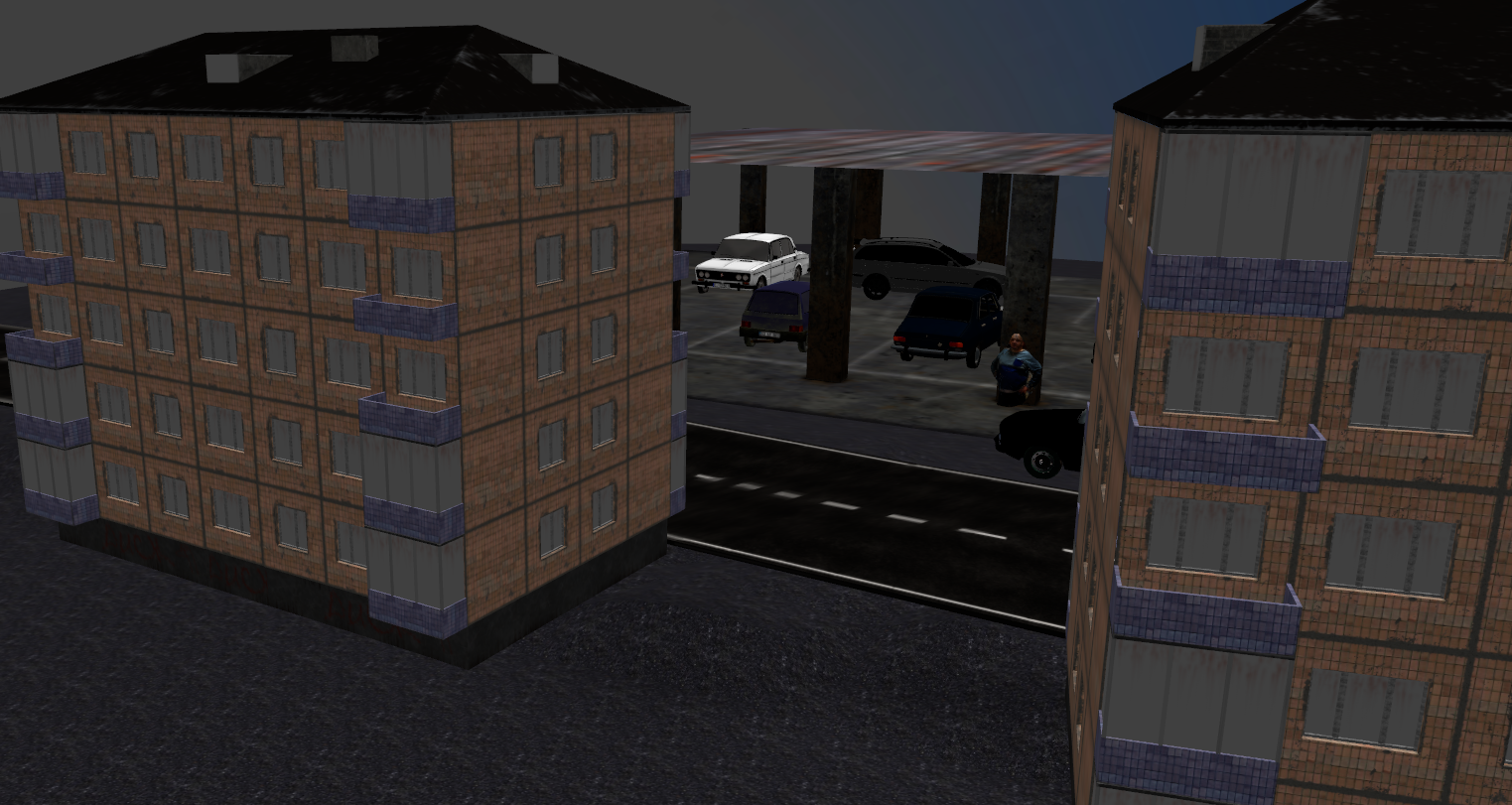




**Ceața**:



**Fără lumină globală:**

****

**Cu lumina locala (punctiforma)**:



# MANUAL DE UTILIZARE

Camera :

* W - deplasare in fata
* S - deplasare in spate
* A - deplasare in stanga
* D - deplasare in dreapta
* R – deplasare in sus
* T – deplasare in jos
* Z – privirea in sus
* X – privirea in jos
* Q – roteste dreapta
* E – roteste stanga

Lumini :

* K – oprire lumina locala
* L - pornire lumina locala
* 6 – pornire lumina globala
* 5 – oprire lumina globala

Vizualizare scena

* C – vizualizarea scenei in modul solid
* B – vizualizarea scenei in modul polygonal
* N – vizualizarea scenei in modul point
* 0 / 9 – porneste / opreste ceata

Animatii:

* F – deschide usa masinii

# CONCLUZII SI DEZVOLTARI ULTERIOARE

Acest proiect a ajutat la dobandirea de noi cunostinte precum utilizarea OpenGL; modelarea, texturarea de obiecte in Blender; gestionarea eficientă a timpului.

În plus, m-a ajutat sa constientizez dificultatea cu care se confrunta dezvoltatorii de jocuri video, precum lipsa de componente hardware potrivite, a memoriei utilizate, dar si de implementarea CORECTA a diferitilor algroritmi.

Ca dezvoltari ulterioare, proiectul ar putea integra functionalitea de a misca/conduce mașinile din scena, dar si intrarea in diferite blocuri din scena. Mai mult, ar putea exista personaje cu care sa poti interactiona.

# REFERINTE

1. <https://free3d.com/> - pentru diferite obiecte
2. <https://www.turbosquid.com/> - pentru diferite obiecte
3. <https://www.cgtrader.com/> - pentru diferite obiecte
4. [Tutoriale Blender –](https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM) Constantin Nandra - pentru bazele blender
5. <https://community.khronos.org> – pentru diferite functii OpenGL