**Documentație de Proiect**

**Craciun Alexandru**  
**Grupa 30237**

**1. Cuprins**

1. Cuprins
2. Specificația subiectului
3. Scenariu  
   3.1. Descrierea scenei și a obiectelor  
   3.2. Funcționalități
4. Detalii de implementare  
   4.1. Funcții și algoritmi speciali  
    4.1.1. Soluții posibile  
    4.1.2. Motivația abordării alese  
   4.2. Model grafic  
   4.3. Structuri de date  
   4.4. Ierarhia claselor
5. Prezentarea interfeței grafice / Manual de utilizare
6. Concluzii și dezvoltări ulterioare
7. Referințe

**2. Specificația subiectului**

Scopul proiectului este crearea unei scene cu obiecte 3D, utilizând bibliotecile prezentate în laborator (OpenGL, GLM, GLFW etc.). Scena creată va include funcționalități precum navigare, lumini, efecte speciale și animații, care vor fi discutate și prezentate în laboratoarele de Procesare Grafică. Utilizatorul trebuie să poată controla scena folosind tastatura sau mouse-ul. Scena trebuie să permită încărcarea obiectelor.

**3. Scenariu**

**3.1. Descrierea scenei și a obiectelor**

Scena proiectului reprezintă o intersecție a două drumuri. De-a lungul drumurilor sunt plasate câteva case, iar pe fiecare drum circulă câte o mașină. În centrul intersecției se află un personaj, care servește ca punct focal al scenei. De asemenea, deasupra intersecției există un balon cu aer cald, care efectuează o mișcare circulară și fluctuează pe verticală conform funcției sinusoidale. În cele patru colțuri ale intersecției sunt instalate felinare, fiecare echipate cu o lumină spot, oferind astfel iluminare punctuală și creând efecte de umbre realiste în scenă. Texturile au fost aplicate obiectelor folosind Blender.

**3.2. Funcționalități**

* Utilizatorul poate vizualiza scena folosind tastele **W, A, S, D** pentru a se deplasa înainte, stânga, înapoi și dreapta. Deplasarea în sus și în jos se face cu **Shift** și **Space**.
* Scena poate fi vizualizată în modurile wireframe, solid sau poligonal, prin apăsarea tastei **R**.
* Ceața poate fi ajustata prin utilizarea tasterlor **C** si **X**.
* Apăsând **2**, lumina este activată/dezactivată
* Ploaia se activeaza prin apasarea tastei **P**
* Pentru a face aplicatia fullscreen se poate apasa pe tasta **F**
* Prin apasarea tastei **M** se poate vizualiza shadow map-ul
* Pornirea animatiei de prezentare a scnerei de pe tasta **1**
* Rotirea personajului se face cu tastele **Q** si **E**
* Rotirea cubului luminos se face cu tastele **J** si **L**
* Ajustarea unghiurilor luminilor spot se face cu tastele **8** si **9**

**4. Detalii de implementare**

**4.1. Funcții și algoritmi speciali**

Funcția **initOpenGLWindow()** gestionează inițializarea ferestrei proiectului. Funcțiile **initModels()** și **initShaders()** sunt responsabile pentru încărcarea scenei, a obiectelor și a shader-elor, iar **initUniforms()** gestionează variabilele uniforme pentru transferul datelor din aplicație către shader-e. Funcția **processMovement()** tratează cazurile de apăsare a tastelor, facilitând modificări ale scenei, camerei sau obiectelor. Randarea scenei, în special a SkyBox-ului, este realizată de funcția **RenderScene()**.

În proiect, am utilizat algoritmi de iluminare pentru a obține efecte vizuale realiste. Acești algoritmi au fost implementați pentru iluminare direcțională, punctuală si de tip spot, contribuind la crearea unui ciclu zi-noapte. De asemenea, un algoritm de ceață a fost utilizat pentru a îmbunătăți atmosfera scenei.

**4.1.1. Soluții posibile**

* **Algoritmi de detectare a coliziunilor**: În OpenGL, detectarea coliziunilor reprezintă intersecțiile dintre obiecte într-o scenă 3D. OpenGL nu oferă suport nativ pentru acest lucru, dar este utilizat împreună cu algoritmi implementați de dezvoltatori.
* **Shadow Mapping**: Algoritmul de shadow mapping este folosit pentru a calcula umbrele obiectelor din scenă. Procesul constă în două etape:   
  Generarea shadow map-ului: Se creează o hartă de adâncime (depth map) din perspectiva sursei de lumină, care stochează informații despre distanța fiecărui punct vizibil la lumină.  
  Aplicarea umbrei: În timpul randării scenei, fiecare fragment este comparat cu shadow map-ul pentru a determina dacă este în umbră sau iluminat.
* **Vizualizarea shadow map-ului**: Prin apăsarea tastei M, aplicația intră într-un mod special de randare care afișează shadow map-ul ca o textură alb-negru pe ecran. Acest mod este util pentru debugging și pentru a înțelege cum sunt generate umbrele.
* **Ploaia**: Algoritmul de ploaie este implementat în funcția initRain, care generează aleator pozițiile picăturilor de ploaie folosind o distribuție uniformă. Picăturile sunt randate ca primitive simple, iar mișcarea lor verticală este animată pentru a simula căderea. Pentru realism, sunt utilizate texturi semi-transparente și iluminare dinamică. Activarea și dezactivarea ploii se realizează prin apăsarea tastei P, care modifică un flag în aplicație pentru a iniția sau opri randarea picăturilor.
* **Ceața**: Efectul de ceață este realizat prin interpolarea culorilor între obiectele scenei și un fundal predefinit, utilizând un shader specific. Funcția adjustFogDensity permite modificarea densității ceții prin intermediul tastelor C (pentru a crește densitatea) și X (pentru a o reduce). Parametrul de densitate este transmis către shader prin variabile uniforme, permițând un control în timp real.
* Alte tehnici vizuale posibile: foc, vânt.

**4.1.2. Motivația abordării alese**

În cadrul proiectului, iluminarea scenei a fost concepută pentru a crea o atmosferă realistă și dinamică, prin combinarea mai multor tipuri de surse de lumină.

1. Lumină Direcțională:

Se utilizează o lumină principală, de tip direcțional, pentru a simula lumina solară. Aceasta este definită printr-un vector (lightDir) și este transformata în spațiul camerei folosind o matrice de rotație. Valorile sale și culoarea (lightColor) sunt trimise ca uniformă către shader-ul principal, contribuind la iluminarea globală a scenei.

2. Spotlight-uri:

Sunt integrate patru surse de lumină de tip spotlight, poziționate la colțurile intersecției. Aceste surse sunt folosite pentru a evidenția anumite zone și pentru a crea umbre realiste.

- Fiecare spotlight are o poziție și o direcție de iluminare, care sunt setate prin variabile uniforme (de exemplu, spotLightPos și spotLightDir).

- Unghiurile interne și externe ale spotlight-urilor se pot regla dinamic cu ajutorul tastelor (8 și 9), care modifică valorile de cutoff, controlând astfel intensitatea și raza de acțiune a luminilor.

**4.2. Model grafic**

Obiectele sunt entitățile din cadrul scenei care reprezintă componentele fundamentale ale modelului grafic. Acestea variază de la forme geometrice simple la modele complexe construite din numeroase primitive. Obiectele sunt caracterizate în principal prin vârfurile lor (**vertices**), care definesc punctele ce constituie forma fiecărui obiect.

Camerele stabilesc perspectiva din care este redată scena, fiind definite prin poziția, orientarea și unghiul lor de vizualizare (**field of view**). Vizibilitatea obiectelor din scenă este determinată de poziția și orientarea camerei, în timp ce unghiul de vizualizare controlează câmpul vizual.

Materialele joacă un rol esențial în definirea atributelor vizuale ale obiectelor din scenă. Aceste atribute includ elemente cruciale precum culoarea, textura, luciul și alte proprietăți care influențează direct procesul de randare. Materialele funcționează sub formă de **shader-e**, care sunt programe scurte executate pe placa grafică. Ele aplică aceste caracteristici geometriei obiectelor, contribuind semnificativ la aspectul și realismul scenei randate. Shader-ele permit personalizarea și ajustarea fină a efectelor vizuale, oferind o experiență diversificată și captivantă în grafica computerizată.

Luminile reprezintă sursele de iluminare din scenă, manifestându-se sub diverse forme, cum ar fi lumini punctuale (**point lights**), lumini direcționale (**directional lights**) sau spoturi (**spotlights**). Aceste surse de lumină nu doar că sporesc vizibilitatea obiectelor, ci joacă și un rol esențial în stabilirea atmosferei și a stării de spirit a scenei. Prin proiectarea umbrelor asupra obiectelor, luminile adaugă adâncime și realism, îmbunătățind atractivitatea vizuală generală. Tipul de lumină ales și configurarea sa pot influența semnificativ atmosfera, evidențiind aspectele cheie ale scenei și modelând percepția privitorului.

**Rasterizarea**, un pas crucial în procesul de randare grafică, transformă grafica vectorială sau modelele 3D în imagini raster, formate din pixeli.

### 4.3 Structuri de Date

Pentru a afișa un obiect 3D, sunt necesare cel puțin un **vertex shader** și un **fragment shader**. Mai jos sunt prezentate câteva componente esențiale și biblioteci utilizate frecvent în programarea cu OpenGL:

#### **Vertex Buffer Object (VBO)**

* **Rol**: Este responsabil pentru stocarea și gestionarea eficientă a datelor despre vârfuri direct pe GPU. Acest lucru permite procesarea rapidă a informațiilor despre poziții, culori, texturi și alte atribute ale vârfurilor.

#### **Vertex Array Object (VAO)**

* **Rol**: Gruparea mai multor VBO-uri într-un singur obiect VAO simplifică procesul de specificare a atributelor vârfurilor. Acest lucru reduce complexitatea și îmbunătățește performanța în timpul randării.

#### **Frame Buffer Object (FBO)**

* **Rol**: Permite randarea în afara ecranului (off-screen rendering), ceea ce este util pentru operațiuni precum randarea în texturi sau crearea de efecte speciale.

#### **GLM (OpenGL Mathematics)**

Oferă tipuri și funcții matematice esențiale pentru grafica 3D:

* **Vec3, Vec4**: Tipuri de vectori 3D și 4D, utilizate pentru a reprezenta poziții, direcții și alte date vectoriale.
* **Mat3, Mat4**: Matrici 3x3 și 4x4, folosite pentru transformări (rotații, translații, scalări) și proiecții în grafica 3D.

#### **Model3D**

* **Descriere**: Reprezintă un model 3D care include geometrie, materiale și alte proprietăți. Această abstracție permite organizarea și manipularea obiectelor 3D complexe în scenă.

#### **Camera**

* **Descriere**: Definește punctul de vedere și perspectiva din care este observată scena. Poziția, orientarea și unghiul de vizualizare al camerei determină modul în care este redată scena 3D.

#### **Shader**

* **Descriere**: Este compus dintr-un **vertex shader** și un **fragment shader**, care procesează datele despre vârfuri și fragmente.
* **Informații suplimentare**: Shader-ele joacă un rol crucial în definirea aspectului vizual al obiectelor, gestionând iluminarea, umbrirea și alte aspecte ale randării.

### 4.4 Ierarhia de Clase

#### **Camera.hpp/Camera.cpp**

* **Funcționalitate**: Această clasă definește comportamentul camerei, incluzând mișcarea și direcția acesteia. Permite ajustarea perspectivei scenei prin poziționarea și orientarea camerei.

#### **Mesh.hpp/Mesh.cpp**

* **Funcționalitate**: Această clasă este responsabilă pentru definirea suprafețelor sau fețelor fiecărui obiect 3D. Reprezintă structura geometrică a obiectelor.

#### **Shader.hpp/Shader.cpp**

* **Funcționalitate**: Gestionează compilarea shader-elor și legarea lor la aplicație. Shader-ele, care sunt responsabile pentru efectele de randare, sunt compilate și organizate în această clasă, facilitând integrarea lor în pipeline-ul grafic.

#### **Skybox.hpp/Skybox.cpp**

* **Funcționalitate**: Această clasă este dedicată creării și afișării skybox-ului în scenă. Implementarea se concentrează pe generarea și randarea skybox-ului, contribuind la atmosfera vizuală generală a mediului.

**5. Prezentarea interfeței grafice / Manual de utilizare**

#### **Rularea aplicației**

Pentru a rula aplicația:

1. Deschide proiectul în **Visual Studio**.
2. Asigură-te că bibliotecile necesare, cum ar fi **GLM** și **GLFW**, sunt corect configurate și importate în proiect.
   * Verifică setările proiectului pentru a include directoarele corecte de headere și biblioteci.
3. Selectează modul de compilare (Debug sau Release) și apasă **Run** pentru a lansa aplicația.

#### ****Controale ale aplicației****

* **W, A, S, D**: Deplasare înainte, stânga, înapoi, dreapta.
* **SPACE, SHIFT**: Deplasare în sus și în jos.
* **R**: Schimbare moduri de vizualizare (wireframe, solid, poligonal).
* **2**: Activare/dezactivare lumină.
* **P**: Activare/dezactivare efect de ploaie.
* **C, X**: Ajustare nivel de ceață (mai densă/mai puțin densă).
* **F**: Activează/dezactivează modul fullscreen.
* **1**: Pornește animația de prezentare a scenei.
* **M**: Activează/dezactivează vizualizarea shadow map-ului.
* **Q si E**: Rotire personaj.
* **J si L**: Rotire cub luminos.
* **8 si 9**: Ajustare unghiului lumnilor spot.

**6. Concluzii și dezvoltări ulterioare**

În concluzie, prin intermediul acestui proiect, am reușit să aplic cunoștințe dobândite în cadrul sesiunilor de laborator. De asemenea, am lucrat cu un instrument nou, **Blender**, pentru a modela scena, a adăuga (sau chiar a crea) diverse texturi pentru obiecte. Ulterior, am importat toate acestea în fișierul proiectului, utilizând funcții specifice din bibliotecile furnizate și afișând scena împreună cu anumite obiecte în aplicație.

Ca rezultat, am dobândit o înțelegere mai profundă a modului în care placa grafică procesează datele pentru a afișa ceea ce vedem pe ecranul computerului. Cu siguranță, există loc pentru îmbunătățiri în cadrul proiectului. Fotorealismul scenei ar putea fi îmbunătățit prin includerea unor obiecte mai diverse.

**7. Referințe**

* [GLFW Documentation](https://www.glfw.org/documentation.html)
* [GLEW](http://glew.sourceforge.net/)
* [Free3D](https://free3d.com/)
* [CGTrader](https://www.cgtrader.com/)