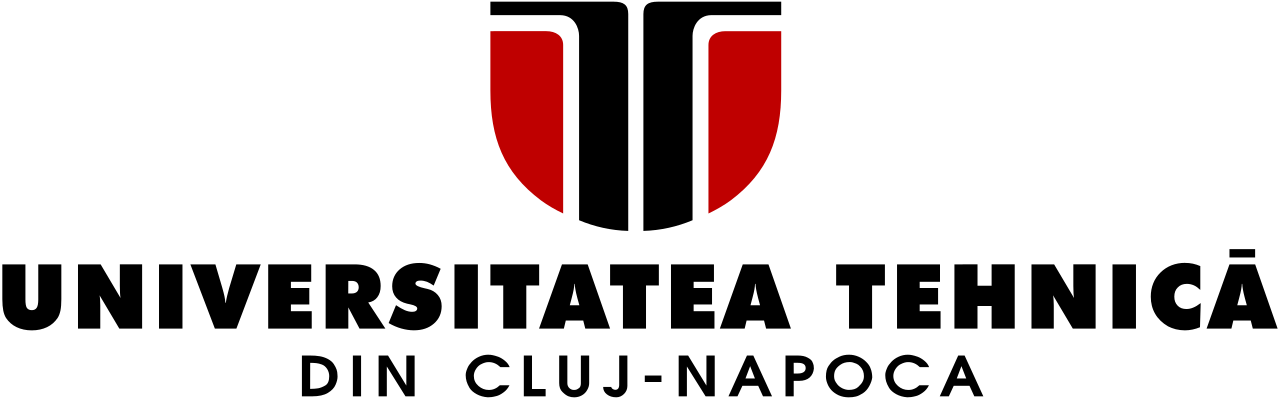
DOCUMENTATIE

Proiect OpenGL cu GLEW si GLFW

****

**NUME STUDENT: Szekely Gergo**

**GRUPA: 30237**

# Cuprins

Contents

[DOCUMENTATIE 1](#_Toc155375606)

[1. Cuprins 2](#_Toc155375608)

[2. Prezentarea temei 2](#_Toc155375609)

[3. Scenariul 2](#_Toc155375610)

[4. Detalii de implementare 2](#_Toc155375611)

[5. Prezentarea interfeței grafice utilizator / manual de utilizare 4](#_Toc155375612)

[6. Concluzii și dezvoltări ulterioare 4](#_Toc155375613)

[7. Referințe 5](#_Toc155375614)

# Prezentarea temei

Proiectul meu folosește OpenGL, împreună cu GLEW și GLFW, pentru a crea o scenă grafică interesantă a unei lumi medievale. Cu limbajul de programare C și mediul de dezvoltare Visual Studio, am creat o experiență vizuală care aduce în viață atmosfera și elementele epocii medievale.

# Scenariul

* 1. **Descrierea scenei și a obiectelor**

Scena proiectului a fost creată în aplicația Blender (aici m-a ajutat tutorialele de la referință și documentația creată de un profesor de la Facultatea de Tehnică din Cluj-Napoca), unde am modelat munți, dealuri și lacuri, aplicând texturi corespunzătoare. Am adăugat diverse obiecte în această scenă medievală. În primul rând, am inclus un castel, format din mai multe obiecte texturate. Ulterior, am creat un mic sat la poalele dealului, în care am plasat mai multe case diferite, texturate în conformitate cu epoca medievală. La baza dealului, am amenajat o piață cu o fântână. La capătul satului, se înalță un templu medieval cu două statui de îngeri, construite de locuitorii săteni. Am introdus în scenă și o moară de vânt, care furnizează făină pentru sat. Pe lac, am plasat două bărci care pleacă în străinătate pentru a aduce alimente pentru sat. În depărtare, am adăugat copaci pentru a forma o pădure..

* 1. **Funcționalități**

Când deschidem aplicația, apare o scurtă animație care ne prezintă peisajul medieval. După ce vedem un vultur care zboară deasupra satului, ne putem deplasa în scenă folosind tastatura și mouse-ul. Tastele ne permit mișcarea, iar cu mouse-ul rotim camera în direcțiile dorite. De asemenea, putem face zoom folosind scroll-ul mouse-ului pentru a ne îndepărta de obiecte sau de scena în ansamblu.În continuare, avem posibilitatea să interacționăm cu iluminarea. Putem ajusta lumina pentru a simula momente diferite ale zilei, inclusiv seara. De asemenea, avem opțiunea de a vizualiza scena în mod punctiform sau de a evidenția liniile din care este construită. În final, avem capacitatea de a adăuga ceață în proiect, dacă dorim.Aceste funcționalități îmbogățesc experiența noastră în explorarea și modificarea scenei medievale

# Detalii de implementare

* 1. **Funcții și algortmi**
     1. **Soluții posibile**

Proiectul este structurat în mai multe funcții. Am funcții pentru **mouseCallBack**, pentru **KeyBordCallBack**, și **windowResizeCallback,** la functiile astea am folosit site-ul learnOpenGl. Am creat aceste funcții pentru ca utilizatorul să poată comunica cu mediul grafic, astfel încât atât mausul, cât și tastatura să fie funcționale. Mai avem și alte funcții, de exemplu, pentru procesarea camerei și pentru gestionarea mișcării.Pe lângă acestea, avem și funcții precum **initModels**, **initShaders**, și **initUniforms**, care ne ajută la inițializarea obiectelor și a shader-elor. După acestea, avem funcțiile **renderScene** și **renderModels**, care generează modelele și construiesc astfel scena noastră. În final, avem **main**, unde apelăm toate funcțiile noastre.Această structură modulară a proiectului permite o gestionare mai eficientă a codului și oferă o organizare clară a funcționalităților.

* + 1. **Motivarea abordării alese**

Am ales această abordare pentru structurarea codului din cauza avantajelor evidente pe care le oferă. Prin organizarea funcțiilor în mod clar și bine definit, codul devine mai ușor de înțeles și de gestionat. Această structură facilitează o colaborare eficientă, iar oricine ar dori să lucreze cu acest cod va avea mai multă claritate asupra funcționalităților și relațiilor dintre ele.Toate funcțiile sunt expuse într-un mod vizibil, ceea ce face posibilă înțelegerea și editarea lor mai ușoară. Această abordare ajută la menținerea unui cod curat și organizat, ceea ce face procesul de dezvoltare și de mentenanță mai eficient.Mai mult, structurarea astfel permite adăugarea facilă a ideilor noi sau extinderea proiectului în viitor. Cu funcțiile bine definite și organizate, este ușor să identifici locurile potrivite pentru a adăuga sau modifica funcționalități, contribuind astfel la scalabilitatea și extensibilitatea proiectului.În general, această abordare a fost adoptată pentru a face codul mai accesibil și pentru a ușura parcursul dezvoltatorilor în explorarea, înțelegerea și extinderea proiectului.

* 1. **Structuri de date**

În proiectul OpenGL, am utilizat vectori și matrici de dimensiuni 2, 3 și 4. Acestea au fost utile în transmiterea shaderului, unde se gestionează iluminarea pentru scena noastră și ne ajută să afișăm obiectele și texturile. De asemenea, au fost esențiale în implementarea funcționalităților camerei, folosindu-le pentru a realiza mișcările necesare în cadrul scenei noastre.Folosirea funcțiilor de glm::rotate și glm::translate s-a dovedit de mare ajutor în manipularea camerei și în animația obiectelor. Aceste funcții permit rotația și translația într-un mod simplu și eficient.Pentru animația inițială, am utilizat funcția **myCamera.getViewMatrix**, care ne furnizează matricea de vizualizare necesară pentru prima animație. În cadrul celei de-a doua animații, am folosit **glm::rotate** pentru a mișca coordonatele obiectului și a-l face să se rotească în continuare.Această abordare a folosirii vectorilor și matricilor nu doar că simplifică codul, dar și îmbunătățește performanța și claritatea în implementarea diferitelor funcționalități ale proiectului.

* 1. **Ierarhia de clase**

În cadrul proiectului, am creat mai multe clase: **Window.cpp**, **tiny\_obj\_loader.cpp**, **std\_image.cpp**, **SkyBox.cpp**, **Shader.cpp**, **Model3D.cpp**, **Mesh.cpp**, **main.cpp**, **Camera.cpp**. Am început lucrul pe clasa **Camera**, unde am completat funcțiile necesare pentru gestionarea camerei, permitându-ne astfel să vizualizăm scena și să mișcăm camera. După finalizarea clasei **Camera**, m-am inspirat din tutorialele de pe YouTube și de pe site-ul oficial al OpenGL pentru a trece la **main.cpp**, unde am completat și introdus funcțiile și ideile necesare pentru a face proiectul funcțional. Am lucrat și cu iluminarea în **Shader** și am găsit soluții la probleme inspirându-mă tot din tutorialele de pe YouTube și de pe LearnOpenGL.Am utilizat și clasa **Model3D.cpp** pentru a crea obiectele și a le plasa în scenă conform intențiilor noastre. Prin intermediul clasei **Shader.cpp**, am stabilit conexiunea cu shader-urile din directorul **Shaders**, care ne ajută în procesul de iluminare a scenei, mișcarea camerei și desenarea acesteia. De asemenea, cu ajutorul clasei **SkyBox.cpp**, am creat conexiunea cu shader-ul **skyBoxShader**, care se ocupă de desenarea cerului care înconjoară scena noastră.Această structură modulară a claselor permite o organizare clară și eficientă a codului. Fiecare clasă îndeplinește un rol specific în proiect, facilitând astfel dezvoltarea, înțelegerea și menținerea codului.

# Prezentarea interfeței grafice utilizator / manual de utilizare

Pentru a naviga în scena noastră, folosim mouse-ul și tastatura. Tastele W, A, S, D ne permit să ne deplasăm înainte, în stânga, înapoi și în dreapta, iar cu mouse-ul rotim scena și implicit camera. Cu tastele Q și R rotim scena, P afișează punctele, L afișează liniile, iar cu N revenim la modul normal al scenei, cu texturi. Tasta F adaugă ceață în scenă, iar G o elimină. La final, cu tasta C rotim lumina pentru a schimba direcția, iar cu Z o ridicăm și cu X o coborâm. Fiți atenți la tasta Z și X, deoarece lumina se mișcă mai încet, evitați apăsările repetate. Și in ultimul rând dacă apăsăm tasta R ni resetează camera la o pozitie predefinită.

# Concluzii și dezvoltări ulterioare

În concluzie, am învățat cum să lucrez în limbajul C cu obiecte, cum să le texturizez și să creez animații cu ele. În diverse etape ale proiectului, am expus și skill-urile mele de a scrie cod în limbajul C, întâmpinând obstacole și rezolvând bucăți de cod care nu doreau să ruleze. Colegii mei și profesorul de la laborator m-au sprijinit, iar împreună am reușit să depășim aceste provocări.În ceea ce privește viitoarele dezvoltări ale proiectului, există mai multe direcții posibile. În primul rând, pot adăuga mai multe animații, de exemplu, pot introduce un dragon pe zidul castelului care suflă flăcări. În al doilea rând, aș putea implementa restricții de trecere prin obiecte, făcând ca utilizatorul să nu poată trece prin ele. În al treilea rând, aș putea să adaug o funcționalitate de ploaie, controlată de la o tastatură. De asemenea, pot implementa iluminare punctiformă și, în ultima parte, pot să fac ca fiecare obiect să aibă propria umbră.Aceste adiții ar putea îmbogăți experiența utilizatorului și ar oferi mai multe posibilități de interacțiune și explorare în cadrul proiectului.

# Referințe

M-am folosit foarte mult de laboratoarele la care am fost si la care am facut laburile.

LeranOpenGL:

<https://learnopengl.com/Lighting/Basic-Lighting>

Lighting:

<https://www.youtube.com/watch?v=ZbszezwNSZU>

<https://www.youtube.com/watch?v=YnBhJbQZLuE>

Camera:

<https://www.youtube.com/watch?v=86_pQCKOIPk>

<https://www.youtube.com/watch?v=q5jOLztcvsM>

Tutorial Blender:

<https://docs.google.com/document/d/1njtWPMmOQNIaD_z9ve8iPRUqQTWdIV_PO-NvPD0nOuM/edit#heading=h.7h28ckb0ku81>

<https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM>