UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

Prelucrare Grafică

Documentație Proiect

Student: Budacă Alexandra

Grupa: 30234

An academic: 2024-2025

Cuprins

1. Prezentarea scenei

Scopul principal al acestui proiect este realizarea unei prezentări fotorealiste a unor scene de obiecte 3D utilizând librăriile OpenGl, GLFW, GLM și mediul de modelare 3D Blender. Utilizatorul trebuie să aibă posibilitatea de a controla scena prin intermediul mouse-ului și a tastaurii.

2. Scenariul

2.1. Descrierea scenei și a obiectelor

Scena realizată surprinde un peisaj a unui sat de la munte, ce oferă o mică fermă de animale, câteva case și “centrul” satului. Pe margine acestuia regăsim un lac cu nuferi, chiar la baza munților, munți care sunt înconjurați de copaci.

Ferma este compusă dintr-un țarc, asamblat în Blender, câteva oi și 2 boi. Fix lângă acest țarc se află 2 căpițe de fân și 2 butoaie, iar puțin mai incolo regăsim 2 grajduri, care au la rândul lor 2 căpițe de fân lângă. De asemenea, în apropiere de acestea putem regăsi si 4 tractoare – unul lângă țarc și trei lângă grajduri.

Satul propriu-zis este compus din 5 case, care au în centrul lor un mic parc. Acest parc este compus din 8 bănci, 4 felinare, 1 fântână și niște tufe. De asemenea, parcul este împrejmuit de un drum de piatra, drum care trece și prin fața fiecărei case, facilitând accesul locatarilor, astfel zona revendicându-și statutul de centrul satului.

La baza munților regăsim un lac acoperit cu nuferi, iar chiar la marginea lacului regăsim 2 bărci cu vâsle – una pe sol si una pe apă.

Pentru efectul de cer am folosit SkyDome, adică o bucată de sfera pe care am texturat-o și aplicat-o ca un capac deasupra scenei mele.

2.2. Funcționalități

→ Pentru navigarea în scena am ales utilizarea tastaturii pentru deplasare (WASD) și utilizarea mouse-ului pentru rotații și apropiere/depărtare de obiect. Astfel, pentru rotații utilizatorul trebuie doar sa miște mouse-ul spre direcția de deplasare dorită, iar pentru scalare trebuie sa dea scroll.
→ Iluminarea scenei este compusă dintr-o lumină direcțională care se poate porni/opri (oprirea ei duce la efectul de întuneric, simulând astfel efectul de soare) și 5 puncte de lumină punctiformă – 4 pentru felinare și una pe fântână, care se pot de asemenea opri/porni.
→ Vizualizarea scenei în modurile wireframe, solid și point, prin apăsarea unei taste.
→ Aplicarea efectelor de ceață, ploaie și tunet la apăsarea unor taste diferite – efectul de ploaie și ceață includ și redarea unui sunet specific.
→ Tur automatizat la apăsarea unei taste, care include și redarea unei melodii
→ Pornire/oprire melodie la apăsarea unei taste pentru modul de navigare în scenă, pentru a descoperi mapa într-un mod mai captivant și mai distractiv.
→ Deplasarea bărcii de pe apa prin intermediul săgeților de la tastatură

3. Detalii de implementare

3.1. Funcții și algoritmi

3.1.1. Soluții posibile

Pentru implementarea proiectului am început prin adăugarea funcțiilor de bază, implementate pe parcursul laboratoarelor:

- *keyboardCallback()* - oferă funcționalitate proiectului prin asocierea unei taste cu o acțiune
- *mouseCallback()* – se ocupă de actualizarea orientării camerei pe baza poziției mouse-ului, pentru a efectua mișcările de rotație
- *processMovement()* – este folosită pentru deplasarea camerei prin intermediul tastelor și vizualizarea scenei în modurile wireframe, solid și point; este diferită față de *keyboardCallback*() deoarece este folosită pentru acțiuni continue, fiind apelată în bucla while din main()
- *initOpenGLWindow()* – configurează și inițializează fereastra pe care va rula aplicația
- *initOpenGLState()* – setează anumite componente pentru buna funcționare a proiectului
- *initObjects()* – încarcă obiectele folosite în scenă
- *initUniforms()*- inițializează variabilele uniform trimise către shader
- *initShaders()* – inițializează shader-urile folosite în proiect

- *renderScene()* – functia care se ocupă cu desenarea și randarea obictelor
- *cleanup()* – eliberează resursele

Ulterior aceste funcții au fost îmbunătățite pentru realizarea funcționalității proiectului final.

Pe lângă aceste funcții de bază, mai avem și funcții extra pentru implementarea de algoritmi diverși: *scrollCallback()* – pentru funcția de zoom in/out, *initPointLight()* – pentru setarea de informații a luminii direcționale, trimise ulterior către shader.

MODELUL DE ILUMINARE

- **Iluminarea direcțională**

  Pentru modelul de iluminare direcțională am implementat iluminarea Phong – calculează culorile finale în fragment shader, folosind normalele interpolate.

  Algoritmul a fost preluat din îndrumătorul de laborator și este implementat în funcția *computeLightDir()* din fragment shader: se calculează direcția luminii, după care se calculează cele 3 componente – ambientala, speculara, difuza.

- **Iluminarea punctiformă**

  Pentru implementarea luminii punctiforme am urmărit tutorialul de pe learnopengl.com.

  Aceasta iluminează radial și uniform în toate direcțiile. Se calculează distanța între poziția luminii punctiforme (bulbLight.position) și poziția fragmentului (fragPos.xyz), apoi se calculează coeficientul de atenuare pentru a reduce intensitatea luminii punctiforme cu distanța.

$$Att = \frac{1.0}{K_c + K_l * d + K_q * d^2}$$

  La final, cele 3 componente – ambientala, difuza, speculara – se inmulțesc cu coeficientul de atenuare și se returnează suma celor trei.

CEAȚA

Pentru realizarea efectului de ceață am ales implementarea propusă la laborator – ceață exponențial pătratică, calculată cu următoarea formulă:

$$fogFactor = e^{-(fragmentDistance * fogDensity)^2}$$

Am definit densitatea ceții, calculăm distanța dintre fragmentul curent și camera, iar apoi aplicăm formula pentru densitate. Astfel vom simula modul în care ceața crește odată cu distanța. La final returnăm un factor de ceață, care variază între 0 (fără ceață) și 1 (cu ceață)

TUNET

Pentru implementarea efectului de tunet creștem de 10 ori intensitatea luminoasă în shader, prin intermediul unui enable trimis ca variabilă uniform și este controlată și de o variabilă uniform timer pentru a simula efectul luminos la un interval de timp stabilit. Acest timer este controlat în programul principal de o variabilă thunderInterval, iar când timer-ul depășește valoarea de interval stabilită, se va reda intensitatea luminoasă puternică și un sunet de tunet.

PLOAIA

Pentru implementarea ploii am importat un obiect picătură în scenă și am generat 10.000 de obiecte de genul pe care le poziționez prin intermediul unui vector de poziție. Aceste picături sunt controlate prin 2 funcții:

- *initRainDrops()* – inițializează picăturile la poziții random
- *updateRaindrops()* – actualizează pozițiile picăturilor cand acestea ajung sub scenă (y negativ)
  La redarea efectului se pornește și sunet de ploaie.

TUR AUTOMAT

Pentru realizarea turului automat mi-am generat 2 fișiere cu coordinate pentru poziția camerei și direcția de vizualizare, folosind un script în Python. În script mi-am declarat 10 puncte cheie pe care vreau să le vizitez, apoi cu ajutorul ecuației dreptei mi-am generat traseul.

Ulterior am modificat Camera.hpp și Camera.cpp, unde am adăugat settere pentru cameraPosition și cameraFrontDirection, iar prin intermediul unui flag, in bucla while din main, citesc pozițiile din fișiere si setez cameraPosition, respectiv cameraFrontDirection

NAVIGAREA CAMEREI

Pentru deplasare folosesc tastatura – în *processMovement()* dacă este apăsată una din tastele WASD, mișc camera în direcția respectiva (înainte, stânga, înapoi, dreapta).

Pentru rotație folosesc mouse-ul prin intermediul funcției *mouseCallback()* care actualizează orientarea camerei pe baza poziției mouse-ului.

Pentru scalare folosesc rotița de la mouse prin intermediul funcției *scrollCallback()*.

GENERARE SUNET

Pentru generarea de sunet am folosit librăria irrKlang pe care am importat-o în Visual Studio, apoi pentru a putea fi folosită am creat funcția *initEngine()* în care am inițializat engine-ul, pentru a putea ulterior încărca sunete. Pe urmă, sunetele au fost încărcate prin intermediul funcției play2D, și a fost folosită de câte ori am importat un sunet nou.

ANIMAȚIA BĂRCII

Pentru animația bărcii am importat obiectul separat de scenă, deja scalat din Blender, iar apoi in funcția *renderScene()* am translatat obiectul la poziția dorită și l-am rotit cu 25 grade pe z. Pentru controlul deplasării bărcii folosim săgețile de la tastaură, iar în funcția *processMovement()* actualizez direcția pe axa z dacă merg înainte/înapoi și pe x dacă merg stânga/dreapta, astfel pot deplasa barca în paralel cu camera.

3.1.2. Motivarea abordării alese

Am ales sa implementez iluminarea Phong și nu Blinn Phong deoarece cel din urmă îmbunătățește relexiile, iar în scena mea nu am obiecte care sa necesite asta. Pentru restul algoritmilor am ales să implementez așa pentru că mi s-a părut cea mai bună alegere pentru proiectul meu.

3.2. Modelul grafic

Pentru alcătuirea scenei am importat mare parte din obiecte de pe internet, în format obj, mai puțin drumul de piatră pe care l-am făcut dintr-un cub aplatizat la care i-am adăugat textură și modifiers – Array, și țarcul pe care l-am asamblat singură. Terenul a fost modelat tot de mine – pentru a adăuga relief – și la fel si lacul – am mai adăugat un plan cu textură. În rest am texturat obiecte întrucât nu veneau doar cu o singură textură, și chiar am separat un obiect – barca am despărțit-o de vâsle pentru a o putea așeza pe pământ și să nu rămână vâslele în aer, iar pentru a doua barcă am mutat vâslele în interiorul ei pentru a o putea plimba pe apă.

3.3. Structuri de date

Pentru implementarea proiectului am folosit biblioteca OpenGL, care vine cu structuri de tip vec<n> și mat<n>, iar în plus am implementat în fragment shader o structură Light cu variabilele aferente luminii punctiforme.

3.4. Ierarhia de clase

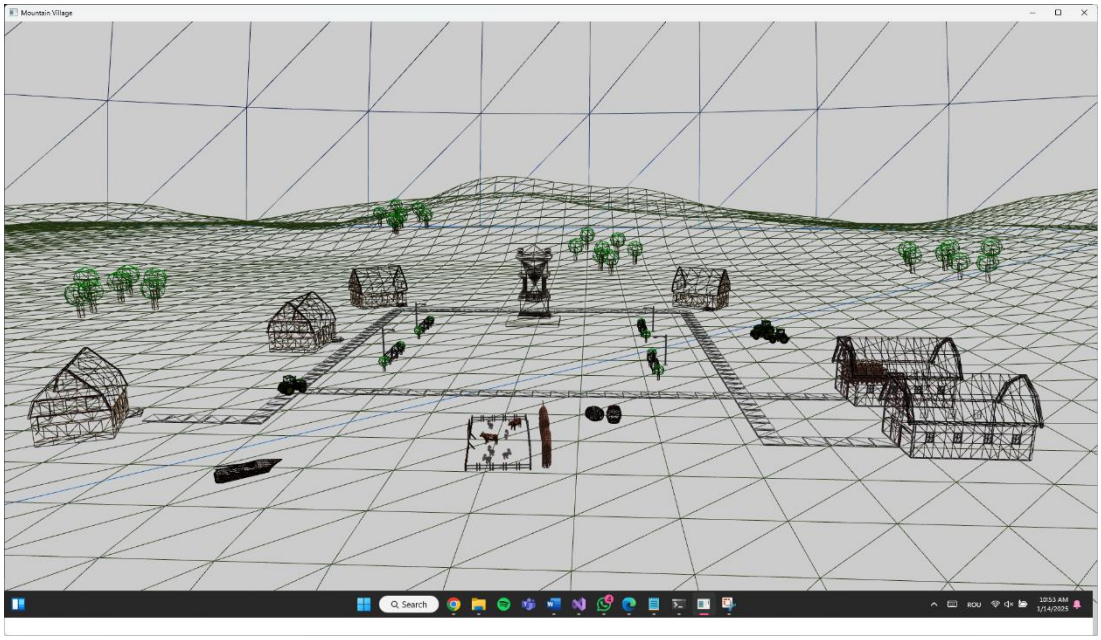
Ierarhia de clase curpinde următoarele headere și .cpp:

- Camera.hpp / Camera.cpp – controlul camerei
- Mesh.hpp / Mesh.cpp – definește vârfurile
- Model3D.hpp / Model3D.cpp – creaza un model 3D
- Shader.hpp / Shader.cpp – încarcă un shader
- std_image.h / std_image.cpp – încarcă imagini (pentru texturi)
- tiny_obj_loader.h / tiny_obj_loader.cpp – încarcă modele .obj
- irrKlang.h – pentru redare sunete
- main.cpp – nucleul aplicației

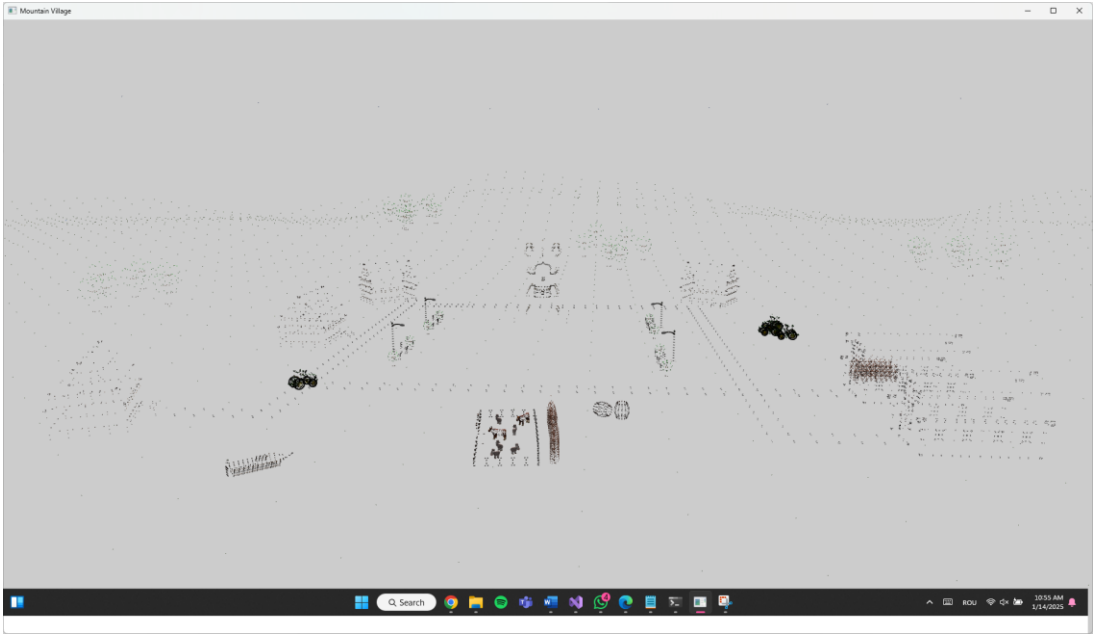
4. Prezentarea interfeței grafice utilizator

MODURI DE VIZUALIZARE

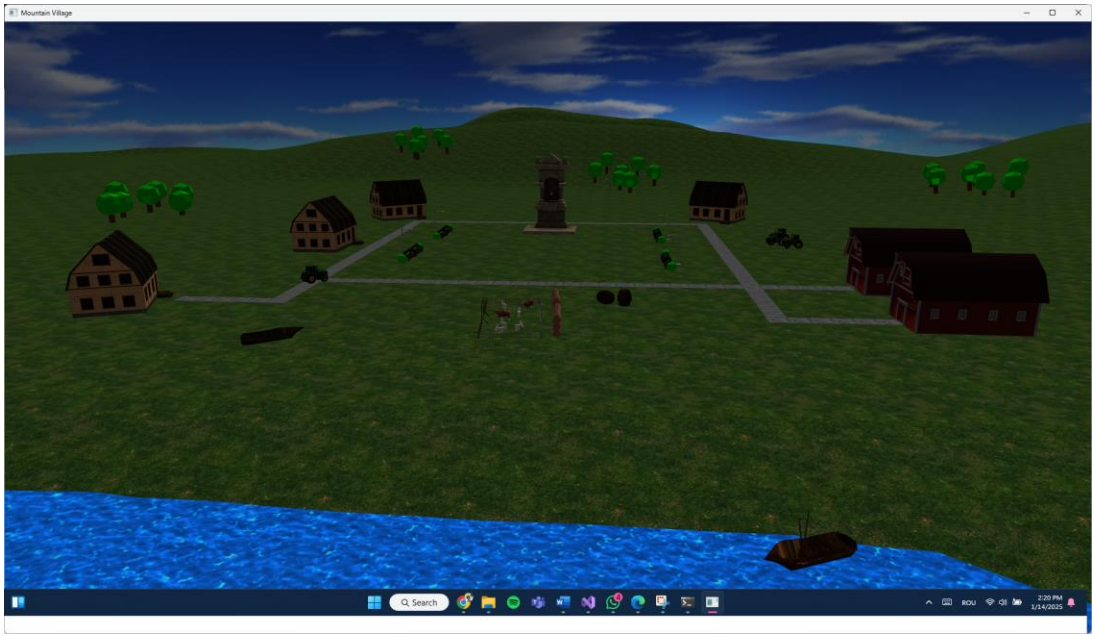
→ **Wireframe**



→ **Point**

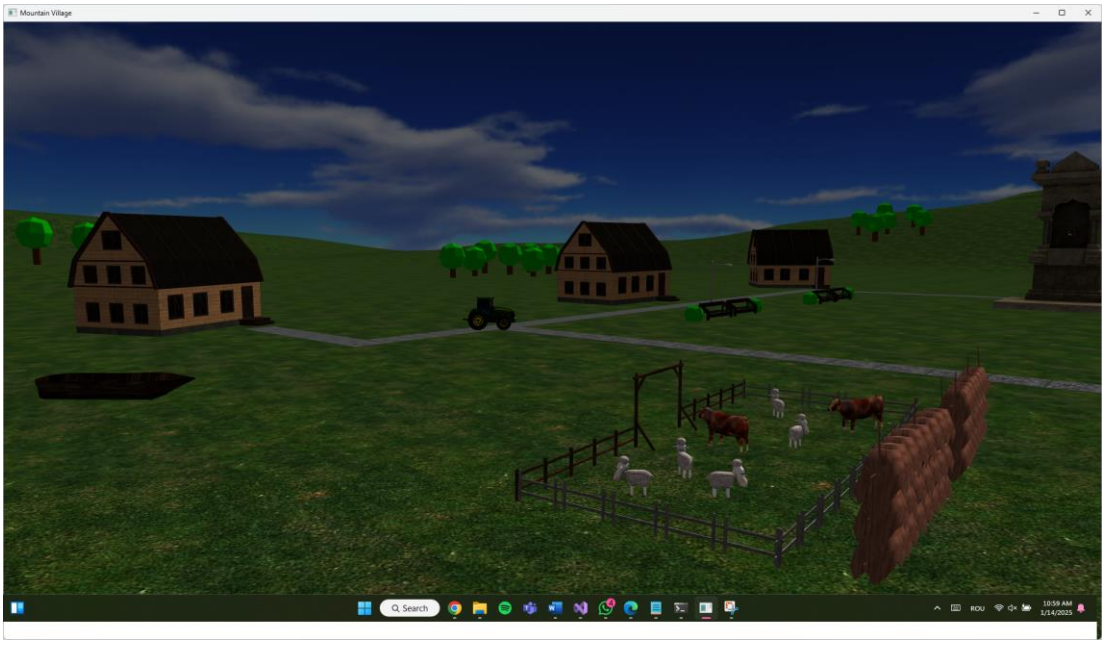


→ **Solid**

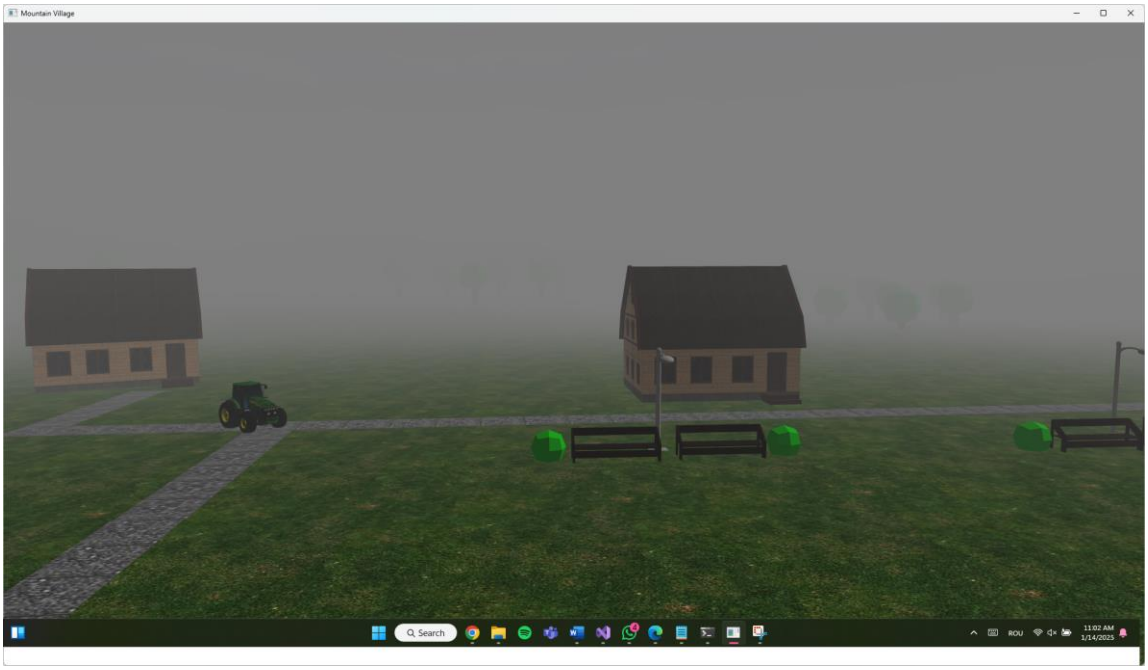


MODURI DE PREZENTARE

→ **Lumină direcțională**

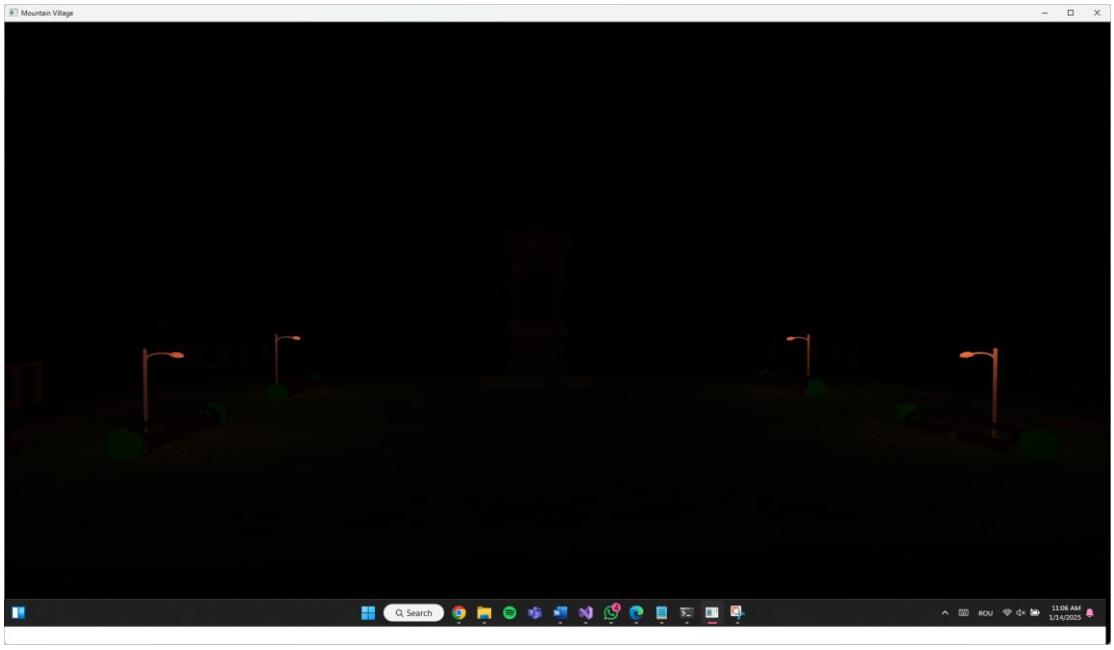


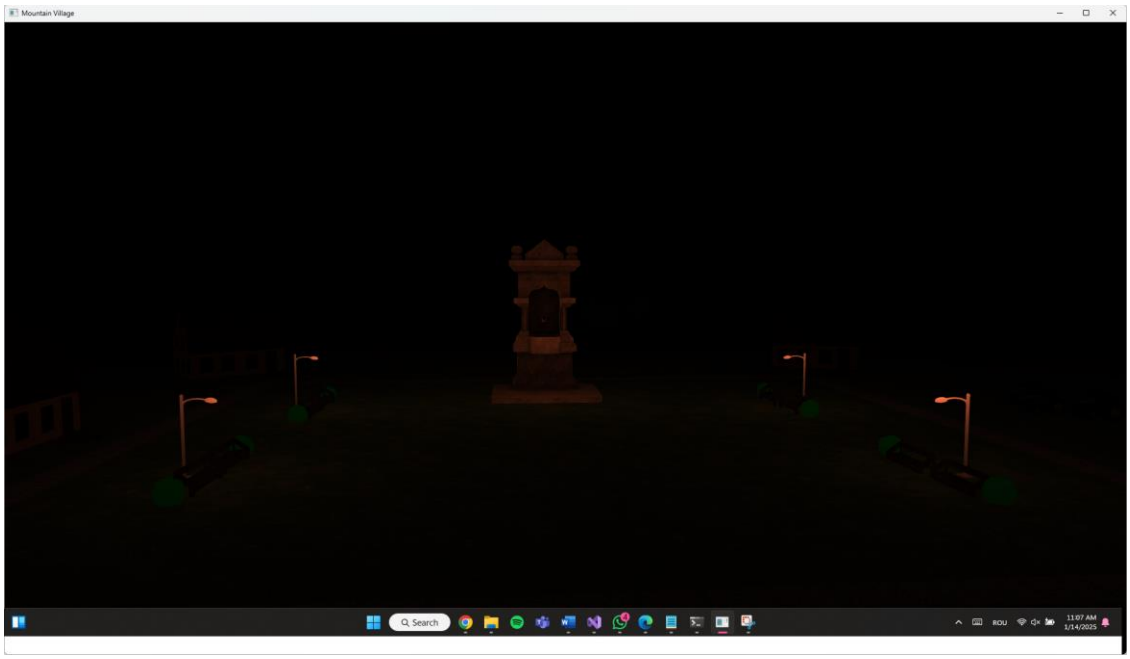
→ **Cu ceață**



→ **Cu ploaie**

→ **Lumină punctiformă**





5. Manual de utilizare

Pentru navigarea în scenă și utilizarea efectelor avem următoarele posibilități:

- W: deplasare înainte
- A: depalsare stânga
- S: depalsare înapoi
- D: depalsare dreapta
- Mișcare mouse: rotire cameră
- Mișcare rotiță mouse (scroll): zoom in / zoom out
- L: pornire/oprire lumină direcțională
- O : pornire/oprire lumină punctiformă fântână
- I: pornire/oprire lumină punctiformă felinare
- R: pornire/oprire ploaie
- T: pornire/oprire tunet
- F: pornire/oprire ceață
- 9 : activare prezentare automată
- 0: dezactivare prezentare automată
- 1 : vizualizare mod wireframe
- 2 : vizualizare mod solid
- 3 : vizualizare mod punctiform
- 8 : activare/dezactivare melodie de fundal
- UP, DOWN, LEFT, RIGHT (săgețile): deplasarea bărcii pe apă

6. Concluzii și dezvoltări ulterioare

În cadrul acestui proiect am reușit să mă familiarizez cu mediul de modelare software 3D Blender și cu pipeline-ul grafic OpenGL. Am aprofundat cunoștiințele învățate la laborator despre lumină, ceață și am reușit să reproduc efectul de tunet de la zero.

Pentru dezvoltări ulterioare, proiectul clar poate fi îmbunătățit prin adăugarea umbrelor, mai multe surse de lumină și implementarea animațiilor pe anumite obiecte, pentru mai mult fotorealism. Alte idei ar fi detecția de coliziune a două obiecte și adăugarea mai multor obiecte în scenă pentru a crea o mapă de joc.

7. Referințe

[1] Directional light. Preluat de pe site-ul learnopengl.com

[2] Camera. Preluat de pe site-ul learnopengl.com

[3] Audio. Preluat de pe site-ul learnopengl.com

[4] Îndrumător de laborator Procesare Grafică. Preluat de pe moodle.cs.utcluj.ro

[5] Water Droplets – Blender Tutorial. Preluat de pe youtube.com