Proiect - Prelucrare Grafică
Universitatea Tehnică Cluj-Napoca

Mesaros Radu

January 2025

Contents

1 Introducere

1.1 Context

Acest proiect prezintă o vizualizare captivantă a unei scene Minecraft, axată pe modul de joc celebru Skyblock, în care jucătorul trebuie să supraviețuiască pe o insulă plutitoare. Scena aduce în prim-plan provocările unice ale acestui mod, capturând esența supraviețuirii și creativității într-un mediu minimalist, dar plin de oportunități.

Insula plutitoare este recreată cu fidelitate, incluzând un copac singular, un bloc de pământ și o ladă cu resurse limitate, care testează ingeniozitatea jucătorului. Dincolo de estetică, proiectul valorifică tehnicile avansate ale graficii OpenGL pentru a reda dinamica mediului, cum ar fi rotația soarelui și a lunii, schimbările atmosferice și efectele de ceață care amplifică sentimentul de izolare în vastitatea cerului.

Elementele de gameplay sunt susținute de shader-e programabile care oferă iluminare realistă și umbre dinamice, subliniind interacțiunea dintre jucător și mediu. Acest proiect nu doar că evidențiază frumusețea simplă și captivantă a lumii Minecraft, dar demonstrează și puterea graficii în timp real pentru a crea experiențe interactive și memorabile. Modul Skyblock prinde viață ca o platformă vibrantă, oferind o provocare constantă pentru jucători de a-și testa limitele creativității și abilităților de supraviețuire.

1.2 Obiective

Obiectivul principal al acestui proiect este de a dezvolta o scenă 3D realistă și interactivă, reprezentând modul de joc celebru Skyblock din Minecraft, utilizând tehnologia OpenGL. Proiectul își propune să exploreze și să demonstreze capacitățile avansate ale OpenGL în redarea graficii în timp real, prin implementarea unor tehnici moderne de iluminare, umbrire și efecte de mediu.

Prin crearea acestei scene, proiectul urmărește să atingă următoarele obiective specifice:

Reproducerea fidelă a mediului Skyblock: Crearea unei insule plutitoare minimaliste, cu elemente iconice precum un copac, o ladă de resurse și blocuri de pământ. Acestea sunt modelate și texturate pentru a reflecta stilul distinctiv al jocului Minecraft.

Implementarea unui sistem de iluminare dinamică: Dezvoltarea unui sistem de iluminare care să includă ciclul zi-noapte, surse de lumină direcționale, precum soarele și luna, și efecte de umbrire realiste prin utilizarea mapării umbrelor.

Integrarea efectelor de mediu: Adăugarea de efecte atmosferice, cum ar fi ceața, ploaia sau variații de lumină, pentru a spori realismul și imersiunea scenei. Aceste efecte sunt realizate prin utilizarea shaderelor programabile și a sistemelor de particule.

Crearea unei experiențe interactive: Permiterea utilizatorilor să exploreze insula Skyblock printr-un sistem de control al camerei.

Optimizarea performanței: Asigurarea unei redări fluide și eficiente a scenei prin implementarea unor tehnici de optimizare, cum ar fi utilizarea eficientă a resurselor grafice, astfel încât să ofere o experiență captivantă și performantă.

Acest proiect aduce la viață atmosfera unică și provocările modulului Skyblock, evidențiind atât frumusețea simplistă a acestui stil de joc, cât și puterea tehnologiilor moderne de grafică.

2 Main.cpp

Descrierea și funcționalitatea fișierului main.cpp Fișierul main.cpp constituie nucleul proiectului, gestionând inițializarea, redarea și interacțiunea scenei 3D în cadrul tematic Minecraft, modul Skyblock. Principalele funcții și componente sunt:

initOpenGLWindow() Inițializează fereastra OpenGL folosind GLFW, setând versiunea OpenGL și configurând contextul grafic.

initOpenGLState() Configurează starea inițială OpenGL, activând testarea adâncimii, culling-ul fețelor și alte setări esențiale pentru redare.

initObjects() Încarcă modelele 3D în scenă, inclusiv insula plutitoare, obiectele interactive (cum ar fi "Creeper") și alte elemente ale mediului.

initShaders() Încarcă și compilează shaderele necesare pentru redarea scenei, incluzând shadere pentru iluminare, efecte speciale și particule.

initUniforms() Inițializează variabilele uniforme utilizate în shadere, cum ar fi matricele de proiecție și vizualizare, direcția și culoarea luminii, dar și parametrii pentru efectele de ceață.

initFBO() Creează și configurează un obiect framebuffer pentru maparea umbrelor, asigurându-se că textura de adâncime este corect configurată.

computeLightSpaceTrMatrix() Calculează matricea de transformare a spațiului de lumină, esențială pentru generarea umbrelor corecte.

drawObjects() Redă obiectele din scenă, aplicând transformările necesare și gestionând atât pasul de adâncime, cât și cel de redare finală.

renderRain() Gestionează și redă efectele dinamice de ploaie prin utilizarea unui sistem de particule.

renderScene() Funcția principală de redare, responsabilă pentru gestionarea întregii scene, inclusiv insula Skyblock, cerul (Skybox), umbrele și particulele.

processMovement() Gestionează mișcarea camerei în funcție de intrările utilizatorului, permițând navigarea în scenă.

keyboardCallback() și **mouseCallback()** Funcții de callback pentru gestionarea intrărilor de la tastatură și mouse, care permit utilizatorului să interacționeze cu scena.

scrollCallback() Gestionează evenimentele de scroll pentru ajustarea câmpului vizual (FOV) al camerei.

updateCreeper() Actualizează animația personajului Creeper, inclusiv mișcarea sa plutitoare.

initSkybox() Inițializează și încarcă texturile pentru cele două Skybox-uri (cer albastru și cer nocturn), cu posibilitatea de a comuta între ele.

updateCameraPresentation() Gestionează modul de prezentare al camerei, deplasând-o automat pe o traiectorie definită.

cleanup() Eliberează resursele alocate, cum ar fi texturile, framebuffer-urile și fereastra OpenGL.

3 Umbrire

Umbre și Tehnici de Umbrire

Umbrele reprezintă un element esențial în crearea unei scene 3D realiste, contribuind la adăugarea profunzimii și contextului vizual. În acest proiect, umbrele sunt generate prin tehnica shadow mapping, care utilizează mai multe etape de procesare grafică pentru a simula modul în care lumina interacționează cu obiectele din scenă.

Generarea Hărților de Adâncime

Procesul de umbrire începe cu generarea unei hărți de adâncime din perspectiva sursei de lumină. Acest pas presupune redarea scenei din punctul de vedere al luminii și capturarea adâncimii fiecărui fragment. Harta de adâncime este crucială pentru determinarea zonelor umbrite, deoarece stochează informații despre distanța fiecărui punct față de sursa de lumină.

Transformarea Coordonatelor

Pentru a aplica umbrele corect, pozițiile vertecsilor sunt transformate în spațiul luminii. Această transformare permite calcularea coordonatelor fiecărui fragment în raport cu sursa de lumină, facilitând identificarea fragmentelor care se află în umbră. Este o etapă esențială pentru a asigura proiecția corectă a umbrelor pe suprafețele obiectelor.

Aplicarea Umbrelor în Scenă

După generarea hărții de adâncime, umbrele sunt aplicate în scenă în timpul pasului de redare principal. În această etapă, fiecare fragment este evaluat pentru a determina dacă este umbrit sau nu. Procesul presupune compararea adâncimii fragmentului curent cu valoarea corespunzătoare din harta de adâncime. Dacă adâncimea fragmentului este mai mare decât cea din hartă, acesta este considerat umbrit.

Calculul Umbrelor

Calculul umbrelor implică utilizarea coordonatelor transformate ale fragmentului și a datelor din harta de adâncime. Pentru a îmbunătăți calitatea vizuală, se aplică tehnici de filtrare, precum Percentage Closer Filtering (PCF). Această metodă presupune eșantionarea mai multor puncte din jurul coordonatelor fragmentului și calcularea unei medii ponderate, ceea ce netezește umbrele și reduce efectele de aliasing.

Integrarea Umbrelor în Iluminare

Umbrele sunt integrate în modelul de iluminare al scenei, influențând atât componentele de iluminare difuză, cât și cele speculare. Prin ajustarea intensității luminii în funcție de prezența umbrelor, scena capătă un aspect mai realist, evidențiind interacțiunea complexă dintre lumină și obiecte.

Această abordare bazată pe shadow mapping asigură umbre dinamice și realiste, contribuind semnificativ la atmosfera și imersivitatea scenei 3D.

4 Ceață

Ceața Ceața este implementată în shaderul de fragment (shaderStart.frag) și se bazează pe calcularea distanței fragmentelor față de camera de vizualizare. Implementarea cuprinde următoarele componente principale:

• Variabile Uniforme: fogColor: Definește culoarea cetii (gri, vec3(0.5f, 0.5f, 0.5f)) fogDensity: Controlează densitatea cetii (0.1f) fogType: Determină tipul de ceață aplicat (0 = linear, 1 = exponential, 2 = exponential squared)

• Calculul Factorului de Ceață: Funcția computeFog() calculează intensitatea cetii pentru fiecare fragment bazându-se pe: Distanța fragmentului față de cameră (length(fPosEye)) Tipul de ceață selectat Pentru ceața liniară (fogType = 0), se folosesc punctele de start (2.0f) și sfârșit (10.0f) Pentru ceața exponențială, se aplică o funcție exponențială care crește cu distanța

• Aplicarea Efectului: Factorul de ceață este clamped între 0.0 și 1.0

Se folosește funcția mix() pentru a interpola între culoarea scenei și culoarea cetii Rezultatul final este stocat în fColor, producând tranziția graduală spre culoarea cetii pe măsură ce obiectele se îndepărtează de cameră Efectul de ceață ajută la crearea unei atmosfere mai realiste și oferă un indiciu vizual despre adâncimea scenei.

5 Ploaie

Ploaie Sistemul de ploaie este implementat folosind un sistem de particule care simulează picături de ploaie. Implementarea constă în următoarele componente principale:

• Structura de Date:

• Variabile de Control:

• Inițializarea Sistemului (initSnow):

Generează 20,000 de particule Setează poziții aleatorii într-o zonă definită Inițializează buffere OpenGL (VAO, VBO) Configurează atributele pentru poziție și viteză

• Shaderul de Vertex (rain.vert): Calculează poziția dinamică a particulelor Aplică efecte de mișcare și vânt Setează dimensiunea picăturilor (glPointSize = 2.0) Calculează transparența bazată pe distanță

• Shaderul de Fragment (rain.frag): Creează picături rotunde folosind gl-PointCoord Aplică estompare la margini folosind smoothstep Setează culoarea și transparența finală

• Randarea (renderRain): Actualizează pozițiile particulelor Aplică efecte de mișcare și gravitație Gestionează resetarea particulelor când ating solul Activează blending pentru transparență Randează particulele ca GLPOINTS

• Control: Activare/dezactivare cu tasta 'P' Când este activată, setează windStrength = 2.0f Când este dezactivată, setează windStrength = 0.0f Sistemul creează un efect realist de ploaie cu particule care cad natural și sunt influențate de vânt, cu transparență și efecte de estompare pentru un aspect vizual convingător.

6 Skybox

Skybox-ul de zi folosește dayLightColor pentru o iluminare mai puternică și caldă Skybox-ul de noapte folosește nightLightColor pentru o iluminare mai slabă și rece Tranziția între acestea afectează întreaga atmosferă a scenei

• Implementare Tehnică: Folosește clasa SkyBox pentru încărcarea și gestionarea texturilor Utilizează shadere specializate pentru randarea corectă Se asigură că skybox-ul este randat primul în scenă Menține skybox-ul centrat pe cameră prin eliminarea componentei de translație din matricea view Sistemul permite o tranziție fluidă între atmosfera de zi și cea de noapte, contribuind la dinamica vizuală a scenei și oferind un context ambiental adecvat pentru restul elementelor 3D.

7 Animații

7.1 Animația creeperului

Animația Creeper - Creeper-ul plutește sus-jos într-o mișcare sinusoidală Amplitudinea de 0.09 unități determină cât de sus/jos se mișcă Viteza de 2.0 controlează frecvența oscilației - Se folosește funcția sinus pentru a crea o mișcare fluidă Timpul este multiplicat cu floatSpeed pentru a controla viteza Rezultatul este înmulțit cu amplitudinea pentru a controla înălțimea - Creeper-ul își menține poziția fixă în coordonatele specificate Doar componenta Y este modificată de animația de plutire Efectul final este că creeper-ul plutește constant sus-jos într-o mișcare lină și continuă, menținându-și poziția în scenă, creând o animație subtilă dar dinamică care adaugă viață personajului.

8 Interacțiune cu scena

Interacțiune cu Scena Controlul Camerei

- W, A, S, D: Taste pentru mișcarea camerei în scenă
- **Moduri de Redare**
- 1-4: Schimbă modul de randare al scenei
- **Efecte de Mediu**
- P: Activează/dezactivează efectul de ninsoare

- O: Comută între skybox-ul de zi și cel de noapte
- **Controlul Luminii**
- J și L: Rotește sursa de lumină
- Moduri de Vizualizare
- M: Comută vizualizarea hărții de adâncime (depth map)
- F: Comută între modul fullscreen și windowed
- **Prezentare și Control**
- C: Activează modul de prezentare automată a scenei
- ESC: Închide aplicația
- Control Mouse
- Mișcare Mouse: Controlează rotația camerei
- Scroll Mouse: Controlează zoom-ul (FOV)
- Acest sistem de control oferă utilizatorului o gamă completă de interacțiuni cu scena, de la navigare și vizualizare până la controlul efectelor de mediu și al modurilor de randare.

9 Concluzii

Proiectul a demonstrat cu succes integrarea Blender și OpenGL pentru a crea o scenă captivantă din Minecraft, evidențiind realismul prin efecte avansate de iluminare, umbre și particule. Interactivitatea și dinamica scenei, cum ar fi ciclul zi-noapte și animațiile personajelor, au oferit o experiență imersivă.

Optimizarea resurselor și utilizarea shaderelor programabile au asigurat performanță ridicată, iar structura modulară permite extinderea proiectului. Acest proiect evidențiază versatilitatea OpenGL și fidelitatea artistică a universului Minecraft.

10 Capturi de ecran

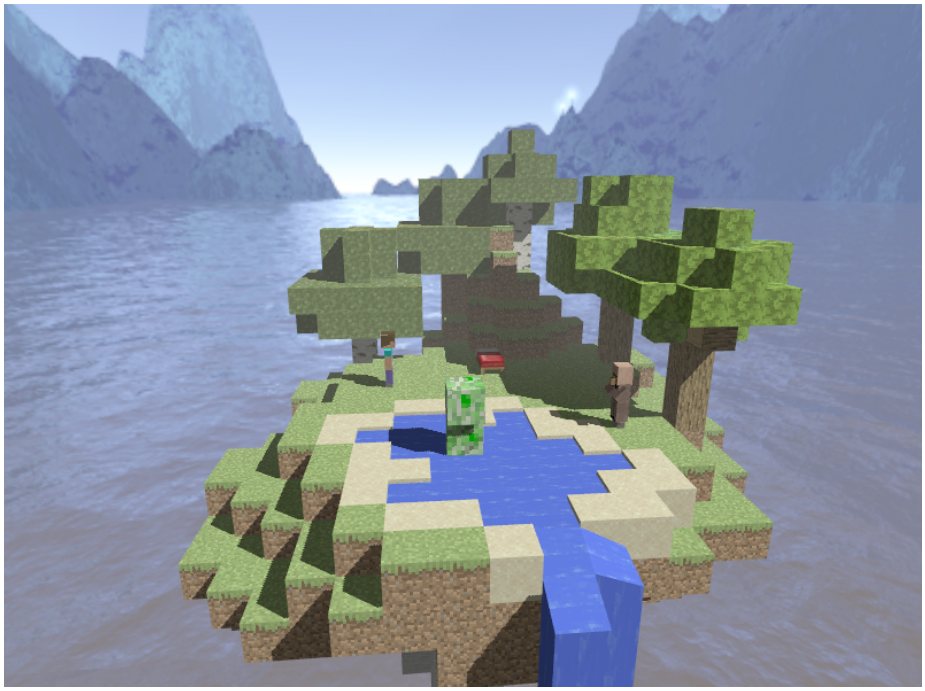

Figure 1: Scena

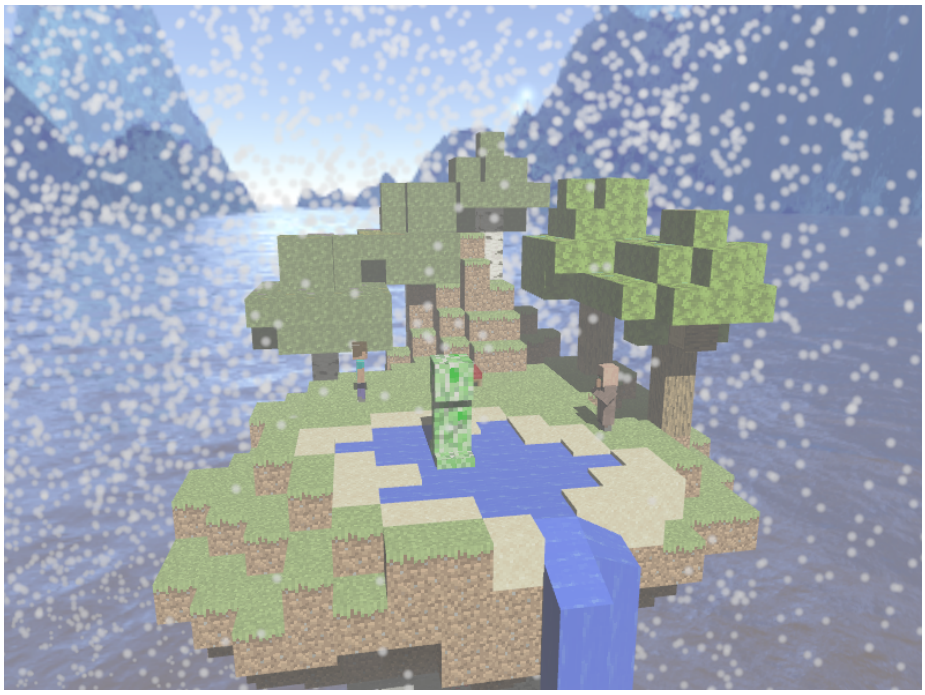

Figure 2: Ceata + Zapada

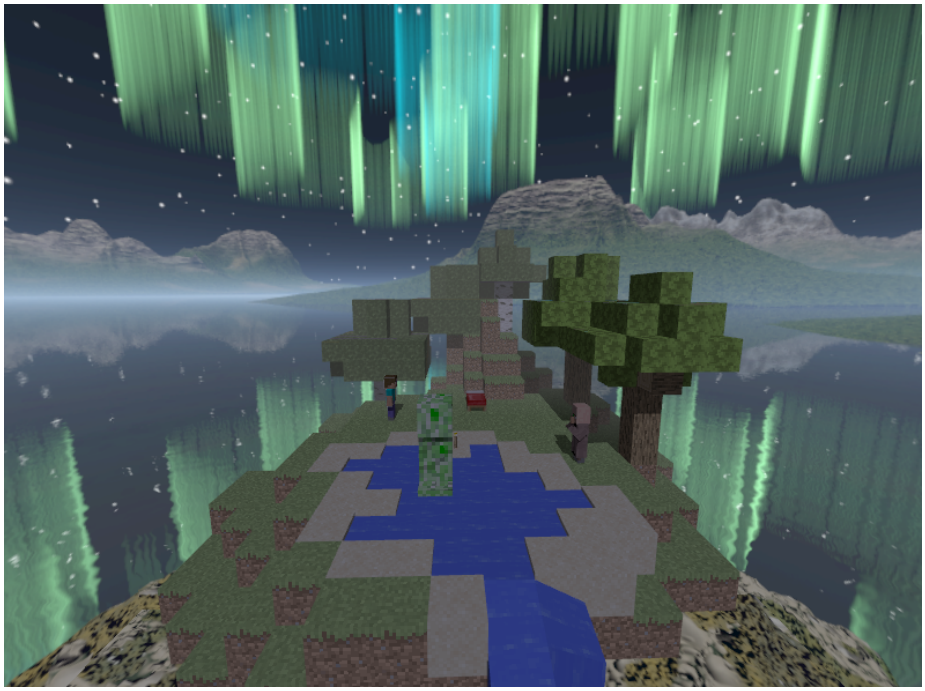

Figure 3: Skybox noapte + schimbare FOV