UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Scena unui concert realizată în OpenGl

Prelucrare Grafică

Autor: Cotei Ruxandra-Maria
Grupa: 30238

FACULTATEA DE AUTOMATICA
SI CALCULATOARE

13 Ianuarie 2025

Cuprins

1 Prezentarea temei

Tema proiectului presupune crearea unei scene interactive 3D ce reprezintă un concert în aer liber, unde utilizatorul este protagonistul cu ajutorul unui First Person Camera FOV, având posibilitatea de a naviga în jurul scenei. Aceasta include un public, echipamente audio, boxe, două corpuri de iluminat stradale și o scenă pe care se află utilizatorul, acesta fiind obiectul principal ce interacționează cu mediul.

Mediul este interactiv, utilizatorul având astfel libertatea de a explora scena din diverse unghiuri. Sursa principală de lumină este soarele, care ilumineză scena într-o manieră constantă, iar utilizatorul poate observa efectele acestei surse asupra obiectelor din jurul său.

Prin urmare, tema proiectului presupune dezvoltarea unei aplicații interactive 3D ce simulează un concert, în care utilizatorul poate explora scena din diverse perspective, iar iluminarea și aspectul obiectelor sunt influențate de sursa de lumină fixă (soarele).

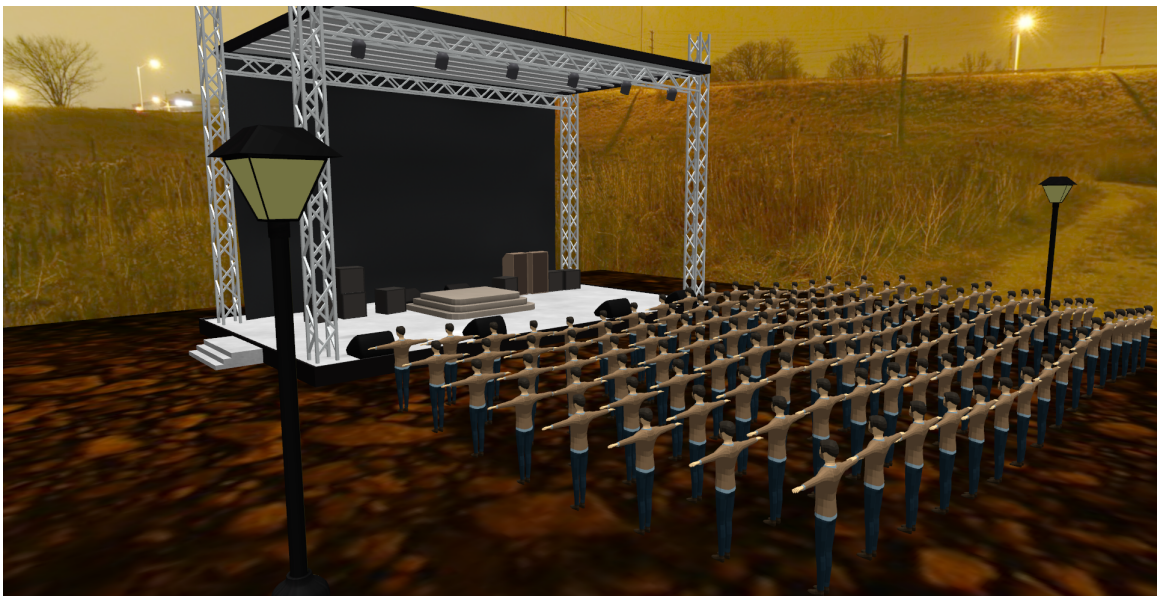

Figura 1: Prezentare scenă

2 Scenariul

2.1 Descrierea scenei și a obiectelor

Scena reprezintă un spațiu de concert în aer liber, având la bază o scenă pe care se află utilizatorul, care joacă rolul principal al acestui eveniment. În jurul său se află un public ce oferă o atmosferă reală de concert. Pe lângă public, scena include echipamente audio, boxe amplasate pe margine, dar și două corpuri de iluminat stradale care completează aspectul iluminării ambientale.

Iluminarea scenei este determinată în principal de soare, care reprezintă sursa de lumină principală. De asemenea, în cadrul scenei sunt implementate diverse efecte vizuale care adaugă realism, precum mișcarea luminii pe măsură ce noaptea se apropie.

2.2 Funcționalități

Aplicația interactivă 3D oferă mai multe opțiuni de interacțiune cu scena, care sunt controlate prin tastatură și mișcarea mouse-ului.

Utilizatorul poate naviga în scenă folosind tastele *WASD* și mișcările mouse-ului pentru a explora mediul din diverse unghiuri și perspective.

Diverse acțiuni se pot efectua cu ajutorul tastelor specifice:

- **Q și E**: Permite utilizatorului să rotească întreaga scenă pe axa verticală, ca și cum utilizatorul s-ar învârti în aer.
- **J și L**: Aceste taste controlează mișcarea soarelui, simulând efectul de înserare prin rotirea acestuia pe axa orizontală.
- **N**: Activează și dezactivează modul de noapte, schimbând iluminarea scenei.
- **F**: Permite activarea și dezactivarea efectului de ceață (fog), adăugând realism scenei.
- **P**: Pornirea și oprirea animației, care poate fi folosită pentru a genera efecte de mișcare sau alte tipuri de animație.

De asemenea, utilizatorul poate selecta diferite moduri de vizualizare a scenei:

- **Modul Solid**: Obiectele sunt vizualizate cu suprafețele complete.
- **Modul Wireframe**: Obiectele sunt vizualizate doar cu structura lor de linii, fără suprafețele complete[5].

3 Detalii de Implementare

3.1 Funcții și Algoritmi

Modelul de Iluminare Phong

Unul dintre modelele fundamentale utilizate pentru iluminare în grafica 3D este modelul de iluminare Phong. Acesta descrie cum lumina interacționează cu suprafețele obiectelor 3D, luând în considerare trei componente principale: iluminarea ambientală, difuză și specular.

Iluminare Ambientală: Această componentă reprezintă iluminarea globală, uniformă, aplicată tuturor suprafețelor într-o scenă. Este independentă de direcția luminii și de unghiul de vizualizare al camerei. Aceasta este esențială pentru a asigura o iluminare de bază, chiar și în zonele unde nu ajunge lumina directă.

Iluminare Difuză: Iluminarea difuză este influențată de unghiul dintre direcția luminii și normalul suprafeței obiectului. Cu cât suprafața este mai apropiată de perpendiculara față de direcția luminii, cu atât iluminarea difuză va fi mai intensă. Acest tip de iluminare produce o dispersie uniformă a luminii pe întreaga suprafață.

Iluminare Specular: Iluminarea specular ține cont de unghiul dintre reflexia luminii și direcția de vizualizare a camerei. Aceasta generează reflexii de tip specular, care sunt vizibile atunci când suprafața este lucioasă și direcția de vizualizare coincide cu unghiul de reflexie al luminii.

Optimizarea modelului Phong: Modelul Blinn-Phong

În implementare am utilizat o variantă mai eficientă a modelului de iluminare Phong, cunoscută sub numele de modelul Blinn-Phong [**hughes2013computer**]. Diferența principală între cele două modele constă în faptul că, în modelul clasic Phong, calculul reflexiei luminii este relativ costisitor din punct de vedere al performanței, deoarece implică calcularea unui vector de reflexie (care necesită în mod normal operațiuni de înmulțire și normalizare a vectorilor pentru fiecare pixel). În schimb, modelul Blinn Phong simplifică acest proces printr-o aproximare care evită necesitatea de a calcula vectorul de reflexie exact, reducând astfel complexitatea calculului. În loc să calculeze exact vectorul reflexiei, modelul Blinn Phong folosește o aproximație bazată pe un factor de influență al unghiului dintre direcția luminii și direcția vizualizării, ceea ce duce la economisirea resurselor de calcul. Această metodă nu doar că îmbunătățește performanța, dar menține un nivel acceptabil de realism vizual, ceea ce este esențial pentru aplicațiile interactive sau pentru simulările în timp real. Astfel, formula generală pentru iluminare în modelul Blinn Phong devine:

$$I = I_a \cdot K_a + I_d \cdot K_d \cdot (\mathbf{L} \cdot \mathbf{N}) + I_s \cdot K_s \cdot (\mathbf{H} \cdot \mathbf{N})^{\alpha}$$

unde:

- I_a este intensitatea luminii ambientale, calculată ca ambientStrength · lightColor,
- I_d este intensitatea luminii difuze, calculată ca $\max(\mathbf{N} \cdot \mathbf{L}, 0.0)$ · lightColor,
- I_s este intensitatea luminii speculare, calculată ca specularStrength $\cdot \max(\mathbf{N} \cdot \mathbf{H}, 0.0)^{\alpha}$ · lightColor,
- $\mathbf{L}$ este vectorul unitate către sursa de lumină (normalizat),
- $\mathbf{N}$ este vectorul normal al suprafeței (normalizat),
- $\mathbf{H}$ este vectorul „jumătate”, care este normalizat și reprezintă media dintre vectorul luminii și vectorul de vizualizare,
- α este coeficientul de rugozitate, care controlează intensitatea reflexiilor speculare.

Prin utilizarea modelului Blinn Phong, am reușit să obțin un echilibru între eficiența calculului și realismul efectelor de iluminare, ceea ce a permis obținerea unui cadru grafic fluid și interactiv.

Interpolarea și Mișcarea

Un alt algoritm important utilizat în acest proiect este interpolarea pentru animație, în special interpolarea liniară, care permite o tranziție lină între diferite poziții sau stări ale unui obiect. În contextul mișcării camerei sau al altor obiecte din scenă, interpolarea liniară (Lerp) este folosită pentru a asigura o mișcare fluidă și continuă, evitând astfel salturile sau mișcările bruste.

Algoritmul de interpolare liniară pentru două valori A și B este dat de formula:

$$Lerp(A, B, t) = A + t \cdot (B - A)$$

unde:

- A și B sunt valorile inițială și finală.
- t este parametrul de interpolare, care variază între 0 și 1.

Acest algoritm permite modificarea lină a poziției unui obiect pe parcursul unei animații, asigurând o mișcare naturală.

Utilizarea Funcției Sinus pentru Mișcare

Un alt algoritm important care contribuie la animarea scenei este utilizarea funcției sinus pentru a crea mișcări oscilante. Funcțiile trigonometrice, cum ar fi sinusul, sunt foarte utile pentru a crea mișcări periodice și fluide ale obiectelor.

Funcția sinus este folosită pentru a simula efecte precum pulsațiile luminii sau mișcarea ondulantă a unor obiecte. Formula generală pentru o mișcare pe axa y folosind funcția sinus este:

$$y(t) = A \cdot \sin(\omega t + \phi)$$

unde:

- A este amplitudinea mișcării, care determină înălțimea maximă.
- ω este frecvența unghiulară, care controlează viteza cu care se schimbă poziția.
- ϕ este faza, care poate ajusta timpul de întârziere al mișcării.
- t este timpul sau parametrul de animație.

Această formulă permite crearea unor mișcări ondulate, care pot fi folosite, de exemplu, pentru a simula efecte de valuri. Mișcarea sinusoidală este folosită frecvent în grafica 3D pentru a adăuga realism în animarea mișcărilor periodice.

Algoritmi pentru Generarea Umbrelor

Pentru a adăuga un efect de realism în scenă, am încercat să implementez algoritmi pentru generarea umbrelor, esențiali în orice simulare grafică 3D. Una dintre cele mai comune tehnici pentru realizarea umbrelor este metoda de *shadow mapping*[1].

În cadrul acestei tehnici, se creează o hartă de adâncime (shadow map) care reprezintă distanța dintre sursa de lumină și fiecare punct din scenă. Ulterior, pentru fiecare pixel vizibil din cadrul camerei, se compară distanța acestui punct față de harta de adâncime pentru a determina dacă punctul respectiv este umbrit sau iluminat.

Pașii generali ai algoritmului de *shadow mapping* sunt:

1. Generarea unei hărți de adâncime din perspectiva sursei de lumină.
2. Proiectarea fiecărui pixel din scena principală în harta de adâncime.
3. Compararea adâncimii pixelului cu valoarea corespunzătoare din harta de adâncime pentru a determina dacă este umbrit.

Cu toate acestea, deși am încercat să implementez această tehnică în proiectul meu, nu am reușit să o aduc la o variantă funcțională completă. Am întâmpinat dificultăți în calcularea corectă a adâncimii și în

alinierea precisă a hărții de adâncime cu scena principală, ceea ce a dus la erori vizuale și o performanță nesatisfăcătoare.

În ciuda acestor dificultăți, implementarea tehnicii de shadow mapping a fost un pas important în înțelegerea modului în care umbrele pot îmbunătăți realismul într-o scenă 3D și va constitui un obiectiv pentru dezvoltările viitoare ale aplicației.

Algoritmi de Optimizare a Performanței

Pentru a asigura o performanță ridicată în aplicațiile 3D interactive, am implementat mai multe tehnici de optimizare. Una dintre cele mai importante tehnici este *frustum culling*, care presupune determinarea dacă un obiect se află în câmpul vizual al camerei înainte de a-l reda[1].

Algoritmul de *frustum culling* presupune calcularea unui con de vizualizare (frustum) care definește zona 3D vizibilă din scenă.

Obiectele care nu se află în interiorul acestui con sunt excluse din procesul de randare, economisind resursele necesare pentru calculul iluminării și al umbrelor.

Coliziuni și Prevenirea Penetrării Obiectelor

Coliziunile între obiecte și camera din scenă reprezintă un aspect crucial al simulării fizicii în grafica 3D, contribuind semnificativ la realismul interacțiunilor dintre entitățile din mediul virtual. Prevenirea penetrarei obiectelor are un impact direct asupra experienței vizuale, deoarece o interacțiune incorectă între obiecte, cum ar fi obiecte care se traversează sau camera care pătrunde în alte entități, distruge iluzia de realitate. În implementarea coliziunilor, am utilizat o abordare bazată pe calculul poziției viitoare a camerei. Astfel, am simulat mișcarea camerei într-o anumită direcție și am verificat dacă aceasta ar fi intrat în coliziune cu vreun obiect din scenă. Dacă intersecția era detectată, mișcarea camerei a fost blocată, împiedicând-o să avanseze în acea direcție. Această metodă ajută nu doar la prevenirea penetrarei obiectelor, dar contribuie și la fluiditatea și eficiența mișcărilor, păstrând integritatea scenei și asigurând o experiență de navigare mai naturală și mai plăcută pentru utilizator.

Concluzie

Fiecare dintre acești algoritmi joacă un rol esențial în crearea unei experiențe vizuale interactive și realiste. Utilizarea unui model de iluminare Phong permite o simulare detaliată a interacțiunii luminii cu obiectele din scenă, iar tehnicile de interpolare și mișcare sinusoidală contribuie la animațiile fluide ale camerei și obiectelor. Generarea umbrelor prin *shadow mapping* adaugă un strat suplimentar de realism, iar optimizările de performanță asigură o rulare eficientă a aplicației chiar și pe hardware mai modest.

3.1.1 Soluții Posibile

În această secțiune, vom analiza soluțiile implementate pentru coliziunile dintre camera utilizatorului și obiectele din scenă, mișcarea camerei și modificările iluminării prin unghiuri de lumină.

Coliziuni între camera utilizatorului și obiectele din scenă

Coliziunile între camera utilizatorului și obiectele din scenă sunt gestionate prin utilizarea unui algoritm bazat pe *CollisionBox* (volum dreptunghiular). Această abordare este simplă, dar eficientă în termeni de performanță, deoarece verificările de coliziune între două volume sunt rapide și implică doar câteva operații matematice.

Algoritmul de bază urmează pașii:

1. Definirea unui volum de coliziune pentru fiecare obiect din scenă.

2. Verificarea intersecției între camera utilizatorului și fiecare obiect 3D din scenă.

Pentru implementare, în loc să calculez direct intersecția dintre două cuburi, am adoptat o abordare mai flexibilă și mai precisă, în care am evaluat mișcarea camerei într-o direcție dată și am verificat dacă această mișcare ar duce la penetrarea unui obiect.

Algoritmul funcționează astfel:

- În primul rând, se calculează poziția camerei dacă aceasta ar avansa într-o anumită direcție (de exemplu, înainte, înapoi, la stânga sau la dreapta).
- Apoi, se verifică dacă această nouă poziție se află într-o zonă de coliziune cu vreun obiect din scenă. Acest lucru se realizează prin calculul poziției camerei în fiecare dintre cele trei axe (X, Y, Z) și verificarea dacă coordonatele sale se află în interiorul volumului obiectului.
- Dacă nu există nicio coliziune, camera își poate continua mișcarea conform direcției stabilite.
- Dacă există o coliziune, mișcarea camerei este blocată în acea direcție, iar camera rămâne pe loc, prevenind astfel penetrarea obiectului.

Această abordare se bazează pe verificarea potențialelor coliziuni doar atunci când camera se deplasează, ceea ce este eficient din punct de vedere al performanței și oferă un control precis asupra interacțiunii camerei cu obiectele din scenă.

Mișcarea Camerei

Algoritmul pentru mișcarea camerei se bazează pe direcțiile de mișcare (înainte, înapoi, stânga, dreapta) și actualizează poziția camerei în funcție de aceste direcții și de viteza specificată. Mai jos sunt pașii generali ai algoritmului de mișcare a camerei:

1. **Determinarea direcției mișcării**: Mișcarea camerei depinde de vectorii de direcție calculați anterior, respectiv *cameraFrontDirection* și *cameraRightDirection*.
2. **Actualizarea poziției camerei**: În funcție de direcția dorită (spre înainte, înapoi, stânga sau dreapta), poziția camerei este modificată prin adăugarea sau scăderea unui factor multiplicat cu viteza mișcării.
3. **Actualizarea camerei țintă**: După fiecare mișcare, camera țintă (*cameraTarget*) este actualizată pentru a reflecta noua poziție a camerei, având în vedere direcția de vizualizare.

Pentru a adăuga interactivitate suplimentară, algoritmul permite și mișcarea camerei folosind mouse-ul. Atunci când utilizatorul mișcă mouse-ul, acesta controlează rotația camerei în jurul axelor sale x (pitch) și y (yaw).

Algoritmul pentru rotația camerei este implementat prin ajustarea unghiurilor de rotație în funcție de mișcările mouse-ului și actualizarea direcției camerei. Aceasta este realizată prin aplicarea modificărilor asupra unghiurilor *yaw* și *pitch*, actualizând astfel direcțiile de vizualizare ale camerei.

Mișcările mouse-ului permit utilizatorului să ”încarce” unghiurile de rotație, astfel încât mișcările camerei devin mult mai fluide și mai naturale. Aceste modificări sunt reflectate în coordonatele *cameraFrontDirection*, *cameraRightDirection*, și *cameraUpDirection*, care sunt recalculate pe baza noilor unghiuri de rotație.

Iluminarea Scenei

Iluminarea scenei a fost realizată utilizând un model de iluminare bazat pe unghiurile unei surse de lumină. Algoritmul de iluminare aplică efecte de iluminare difuză, speculară și ambientală asupra obiectelor, iar în funcție de poziția sursei de lumină, umbrele sunt generate pe obiectele din scenă. Pașii algoritmului sunt următorii:

1. Definirea unei surse de lumină de tip *direcțională*, cu unghiul de rotație specificat pentru sursa de lumină.
2. Calculul intensității luminii pentru fiecare punct de pe suprafața obiectelor din scenă.

3. Aplicarea modelului de iluminare Blinn-Phong pentru a calcula iluminarea difuză și specular pe baza poziției și direcției sursei de lumină.
4. Generarea umbrelor utilizând tehnica *shadow mapping*, care presupune crearea unei hărți de adâncime pentru a aplica umbrele corect pe obiectele din scenă.

Am implementat o sursă de lumină *direcțională*, care generează lumina pe întreaga scenă dintr-o direcție specifică, contribuind la iluminarea uniformă a obiectelor.

Modelul de iluminare Blinn-Phong este utilizat pentru a calcula iluminarea în trei componente, așa cum au fost explicate și anterior:

- **Iluminare ambientală**;
- **Iluminare difuză**;
- **Iluminare specular**.

Aceste componente sunt combinate pentru a produce o iluminare realistă pe obiectele din scenă. âpar De asemenea, am încercat să implementez și alte tipuri de surse de lumină, cum ar fi *punctiforma* și *spotlight*, dar, din păcate, nu am reușit să le implementez complet și corect în cadrul aplicației.

3.1.2 Motivarea Abordării Alese

Coliziuni cu CollisionBox

Algoritmul de coliziune bazat pe volume dreptunghiulare este ales datorită eficienței sale. Calculul coliziunilor între poziția în care se va afla camera și diverse obiecte dreptunghiular este simplu și implică doar compararea coordonatelor extremelor obiectelor. Acest tip de algoritm este rapid și scalabil pentru aplicațiile interactive, cum ar fi jocurile sau simulările grafice 3D. O abordare mai complexă, cum ar fi testele de coliziune pe baza formei exacte a obiectelor, ar putea adăuga complexitate inutilă și ar reduce performanța.

Mișcarea Camerei

Soluția de mișcare bazată pe tastatură și mouse este aleasă pentru simplitatea și eficiența sa și pentru că este standard în aplicațiile de realitate virtuală și simulările interactive. De asemenea, aceasta este o soluție familiară pentru utilizatori, oferindu-le control total asupra camerei fără a necesita dispozitive de intrare suplimentare.

Iluminarea și Umbrele

Modelul Blinn-Phong este eficient pentru aplicațiile în timp real, oferind un echilibru între calitatea iluminării și performanță. Utilizarea *shadow mapping* pentru generarea umbrelor adaugă realism fără a necesita calcule de lumină foarte complexe.

3.2 Modelul Grafic

Modelele 3D au fost create utilizând Blender [2], un software open-source recunoscut pentru capabilitățile sale avansate de modelare[4].

Implementarea efectelor vizuale s-a bazat pe utilizarea shaderelor, conform abordărilor descrise de Shreiner et al. [**Angel2011**].

Scena 3D este compusă din mai multe obiecte esențiale care creează o atmosferă dinamică și imersivă, inclusiv scena, publicul și echipamentele de iluminat. Aceste obiecte sunt plasate în spațiul 3D folosind o combinație de matrice de transformare care asigură poziționarea corectă și dimensionarea acestora. De asemenea, un skybox este utilizat pentru a crea un fundal infinit, oferind senzația unui mediu vast și continuu, contribuind astfel la crearea unei atmosfere de concert. Împreună, aceste elemente formează un mediu 3D interactiv, unde utilizatorul poate observa efectele vizuale realiste generate de iluminare și umbre.

Iluminarea

Iluminarea din scenă este controlată printr-o sursă de lumină direcțională, care poate fi rotită pentru a simula schimbarea poziției soarelui. Această sursă de lumină direcțională iluminează întreaga scenă, iar algoritmii de iluminare aplică efecte vizuale pe fiecare obiect, generând iluminare difuză, specular și ambientală. Întreaga scenă beneficiază de un iluminat constant din direcția sursei de lumină. Iluminarea este ajustată în funcție de unghiul sursei de lumină (soarele), astfel încât utilizatorul poate manipula unghiul sursei pentru a observa efectele de iluminare și înserare în scenă.

Scena 3D

Scena 3D este creată prin combinarea unor obiecte 3D esențiale, cum ar fi scena propriu-zisă, publicul și soarele. Obiectele sunt redimensionate și plasate corect în spațiul 3D utilizând o serie de matrice de transformare: matrice de modelare pentru a poziționa obiectele, matrice de vizualizare pentru a determina perspectiva camerei și matrice de proiecție pentru a ajusta distanțele și perspectivele scenei. Sursa principală de lumină este simulată printr-o lumină direcțională fixă, precum soarele, care se poate roti pentru a modifica iluminarea scenei, permițând utilizatorului să observe schimbările de intensitate și direcție a luminii în funcție de unghiul sursei.

Simularea unui Concert cu Mulțime

În cadrul simulării unui concert, mulțimea este formată din obiecte 3D plasate la distanțe egale, pentru a reda un public organizat și ordonat. Această distribuție uniformă adaugă un nivel de realism, oferind impresia unei mulțimi compacte și bine organizate.

SkyBox

Skybox-ul reprezintă fundalul scenei și este compus dintr-o serie de imagini texturate care formează un fundal 3D infinit, înconjurând întreaga scenă. Acesta ajută la adăugarea unui efect de adâncire și distanță, contribuind la crearea unei senzații de imersiune completă a [3]. Skybox-ul oferă o iluzie de mediu vast și continuu, completând astfel atmosfera concertului și oferind contextul necesar pentru restul obiectelor din scenă. De exemplu, un skybox cu un peisaj de cer albastru sau o panoramă de oraș poate adăuga un cadru natural pentru concertul 3D.

3.3 Structuri de Date

Pentru a gestiona scena 3D, sunt folosite următoarele structuri de date:

CollisionBox

Structura `CollisionBox` definește obiectele 3D cu care camera poate interacționa. Fiecare obiect are o poziție și dimensiuni definite, iar funcțiile de coliziune permit verificarea contactului între camera utilizatorului și aceste obiecte. Aceasta include un vector `objects` ce stochează obiectele din scenă care pot interacționa cu camera, având un rol esențial în gestionarea coliziunilor și menținerea controlului asupra elementelor din scenă.

SkyBox

Structura `SkyBox` reprezintă fundalul 3D al scenei și conține texturi care formează un fundal vast. Aceste texturi sunt plasate pentru a crea impresia unui spațiu larg și nelimitat, contribuind la senzația de imersiune a utilizatorului. Skybox-ul ajută la stabilirea atmosferei generale a scenei și completează efectele vizuale ale obiectelor din scena 3D.

3.4 Ierarhia de Clase

Ierarhia de clase este una simplă și modulară. Fiecare componentă a scenei este reprezentată de o clasă distinctă, ceea ce permite o întreținere ușoară și o scalabilitate mai bună a aplicației.

Camera

Clasa `Camera` este esențială pentru controlul perspectivei scenei. Ea gestionează mișcările și rotațiile camerei în funcție de intrările utilizatorului. Camera controlează și poziția și direcția vizualizării, influențând interacțiunea utilizatorului cu scena 3D.

SkyBox

Clasa `SkyBox` este responsabilă pentru crearea fundalului din imagini texturate, care oferă senzația unui spațiu vast și deschis. Aceasta ajută la stabilirea atmosferei generale a scenei, contribuind la crearea unei experiențe imersive și a unui fundal 3D nelimitat.

Window

Clasa `Window` gestionează fereastra aplicației, inclusiv redarea scenei și capturarea intrărilor utilizatorului. Aceasta controlează dimensiunea ferestrei, și interacțiunile cu utilizatorul, asigurând astfel o experiență interactivă și fluidă.

Mesh

Clasa `Mesh` reprezintă obiectele geometrice din scenă. Aceasta conține informații despre formele 3D, cum ar fi vertecșii, muchiile și fețele obiectelor. Mesh-ul este esențial pentru reprezentarea obiectelor și a structurilor din scena 3D, având un rol important în calcularea ilustrației vizuale.

Model3D

Clasa `Model3D` este folosită pentru a reprezenta un obiect 3D complex, format dintr-o colecție de meshe-uri. Aceasta include informații despre textura, transformările și animarea obiectului, permițând realizarea unor modele complexe care pot fi incluse în scenă.

4 Prezentarea interfeței grafice utilizator / Manual de utilizare

4.1 Interfața Grafică Utilizator

Interfața grafică a aplicației este concepută pentru a oferi utilizatorului o experiență interactivă și intuitivă. Scena 3D poate fi explorată prin mișcarea camerei și interacțiunea cu obiectele din scenă. Aceasta include:

- **Vizualizare dinamică:** Utilizatorul poate naviga în scenă folosind tastatura și mouse-ul. Mișcarea pe axele X, Y și Z este realizată prin taste, iar rotirea camerei se poate face cu mouse-ul.
- **Controlul luminii:** Utilizatorul poate modifica unghiul sursei de lumină pentru a simula efectele de iluminare pe obiecte. De asemenea, sursa de lumină poate fi dezactivată prin apăsarea tastei **N**.
- **SkyBox:** Fundalul 3D este prezentat printr-un skybox, oferind senzația unui mediu vast și imersiv.
- **Efecte speciale:** Utilizatorul poate activa ceața prin apăsarea tastei **F**, adăugând un efect vizual suplimentar care modifică atmosfera scenei.
- **Animare:** Aplicația include o animație de prezentare a scenei, care poate fi activată prin apăsarea tastei **P**.
- **Interacțiuni cu obiectele din scenă:** Obiectele 3D pot fi vizualizate și explorate pentru a oferi un sentiment de imersiune.

Interfața este optimizată pentru a asigura o utilizare facilă, fără a necesita un efort de învățare semnificativ.

4.2 Manual de utilizare

Pentru a utiliza aplicația, urmați pașii de mai jos:

1. **Lansarea aplicației:** Deschideți aplicația executabilă. Aceasta va deschide fereastra principală, iar scena 3D va fi redată automat.
2. **Navigarea în scenă:**
   - Folosiți tastele **W, A, S, D** pentru mișcarea camerei înainte, înapoi și lateral.
   - Utilizați mouse-ul pentru a roti camera și a vizualiza scena din unghiuri diferite.
3. **Controlul luminii:**
   - Schimbați unghiul luminii folosind tastele **L** și **R**.
   - Dezactivați sursa de lumină apăsând tasta **N**.
4. **Activarea efectelor speciale:**
   - Apăsați tasta **F** pentru a activa efectul de ceață.
   - Apăsați tasta **P** pentru a porni animația de prezentare a scenei.
5. **Vizualizarea fundalului:** Skybox-ul este vizibil automat din orice unghi al camerei.
6. **Închiderea aplicației:** Închideți aplicația apăsând butonul de închidere al ferestrei sau folosind comanda `Esc`.

Sfaturi pentru utilizator

- Ajustați viteza camerei în funcție de preferințele personale, dacă aplicația oferă această opțiune.

Aplicația este compatibilă cu majoritatea tastaturilor și mouse-urilor, fiind optimizată pentru a oferi o experiență fluidă pe o gamă largă de configurații hardware.

5 Concluzii și Dezvoltări ulterioare

Concluzii

Proiectul prezentat a demonstrat o integrare reușită a elementelor de bază necesare pentru construirea unei scene 3D interactive și imersive. Prin utilizarea tehnicilor moderne de iluminare, implementarea unui skybox și posibilitatea de a explora scena cu ajutorul camerei, aplicația oferă utilizatorului o experiență captivantă. De asemenea, structura modulară a codului permite o gestionare ușoară a componentelor și deschide calea către îmbunătățiri viitoare.

Cu toate acestea, există anumite limitări și provocări rămase, cum ar fi implementarea completă a umbrelor și extinderea posibilităților de iluminare. Aceste aspecte oferă un spațiu amplu pentru dezvoltări ulterioare.

Dezvoltări ulterioare Pentru viitor, se propune extinderea proiectului prin următoarele îmbunătățiri și funcționalități:

- **Implementarea corectă și completă a umbrelor:** Dezvoltarea unui sistem mai avansat pentru generarea umbrelor, care să ia în considerare poziția și forma tuturor obiectelor din scenă.
- **Adăugarea luminilor poziționale:** Integrarea surselor de lumină care să radieze dintr-un punct fix, pentru a crea efecte vizuale mai complexe și mai realiste.
- **Implementarea luminilor de tip spotlight:** Crearea unor surse de lumină direcționale cu un unghi limitat de emisie, utile pentru scenarii precum iluminarea de scenă sau focalizarea pe un obiect specific.
- **Vizualizarea în modurile poligonal și smooth:** Oferirea utilizatorului posibilitatea de a schimba modul de afișare a obiectelor între un mod cu muchii ascuțite (poligonal) și unul mai fluid (smooth shading).
- **Adăugarea animațiilor complexe:** Dezvoltarea unor animații suplimentare pentru obiectele din scenă, cum ar fi mișcarea mulțimii sau efectele de mișcare a luminii.
- **Implementarea opțiunilor de personalizare:** Oferirea utilizatorilor unor setări suplimentare, precum ajustarea culorii luminii, intensității sau vitezei camerei.
- **Optimizări pentru performanță:** Reducerea încărcării pe procesor și GPU prin utilizarea unor tehnici avansate de culling și optimizare a geometriei.

Aceste dezvoltări ulterioare nu doar că vor îmbunătăți calitatea vizuală a proiectului, dar vor transforma scena într-o platformă completă și complexă, cu multiple aplicații în domeniul divertismentului sau al simulării 3D.

Referințe

[1] Tomas Akenine-Möller, Eric Haines și Naty Hoffman, *Real-Time Rendering*, a 4-a ed., Boca Raton, FL: CRC Press, 2018.

[2] Blender Foundation, *Blender: The Free and Open Source 3D Creation Suite*, Available at `https://www.blender.org`, 2023.

[3] Anton Gerdelan, *Anton's OpenGL 4 Tutorials*, a 6-a ed., Self-published, 2016.

[4] Cosmin Nandra, *OpenGL Programming Tutorials Playlist*, `https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM`, Accessed: 2025-01-13, 2023.

[5] Joey de Vries, *Learn OpenGL: A step-by-step guide for learning modern OpenGL (4.0 and above)*, Learn OpenGL, 2020.