Proiect prelucrare grafica

2024-2025

Rujac Roxana
Grupa 30234

Cuprins

Prezentarea temei

Tema proiectului a fost realizarea unei scene grafice cu ajutorul aplicatiei Blender si API-ului OpenGL care sa puna in valoare pipeline-ul grafic programabil.

OpenGL este o interfata de programare a aplicatiilor (API) care poate randa grafica vectoriala 2D si 3D. Aceasta interactioneaza cu unitatea de procesare grafica (GPU).

Scenariu

Descrierea scenei si a obiectelor

Tema pe care am ales-o pentru acest proiect este reprezentarea unei scene 3D care surprinde o insula parasita in mijlocul oceanului. Pe aceasta se afla o caracatita si cateva creaturi ce pazesc o comoara. Insula este populata si de obiecte specifice, cum ar fi animale (testoase, pasari, rechini), obiecte de decor (scaun de plaja, minge de plaja, umbrela de plaja, butoaie) si diferite plante (palmieri, diferite tipuri de ierburi).

De asemenea, in scena se afla si doi pirati cu barca lor esuata langa mal care doresc sa fure bogatiile ascunse ale insulei. Un alt aspect ce se poate mentiona este faptul este prezenta ploaia, iar la apasarea unei taste putem avea si ceata.

Descrierea functionalitatilor

Utilizatorul are posibilitatea sa interactioneze cu scena prin intermediul mouse-ului si a tastaturii pentru o experienta mai interactiva.

Acesta poate naviga, poate schimba modul de vizualizare (wireframe, polygonal, smooth), tipul de lumina (directionala sau punctiforma), poate alege sa porneasa sau sa opreasca efectul de ceata sau cel de transparenta al obiectelor. De asemenea, se poate efectua un tur automat al intregii scene la apasarea altei taste.

Detalii de implementare

Functii si algoritmi

In acest proiect am folosit mai multe functii, unele dintre care erau deja implementate. Cele mai importante si notabile functii pe care le-am folosit sunt:

-renderScene()
-drawObjects()
-initUniform()
-initObjects()

In ceea ce priveste algoritmii, am folosit resurse din laboratoare pentru a implementa diferite efecte: ceata, ploaie, trasnparenta, fragment discarding, lumina punctiforma si directionala, umbre.

Solutii posibile & motivarea abordarii alese

1. Ceata

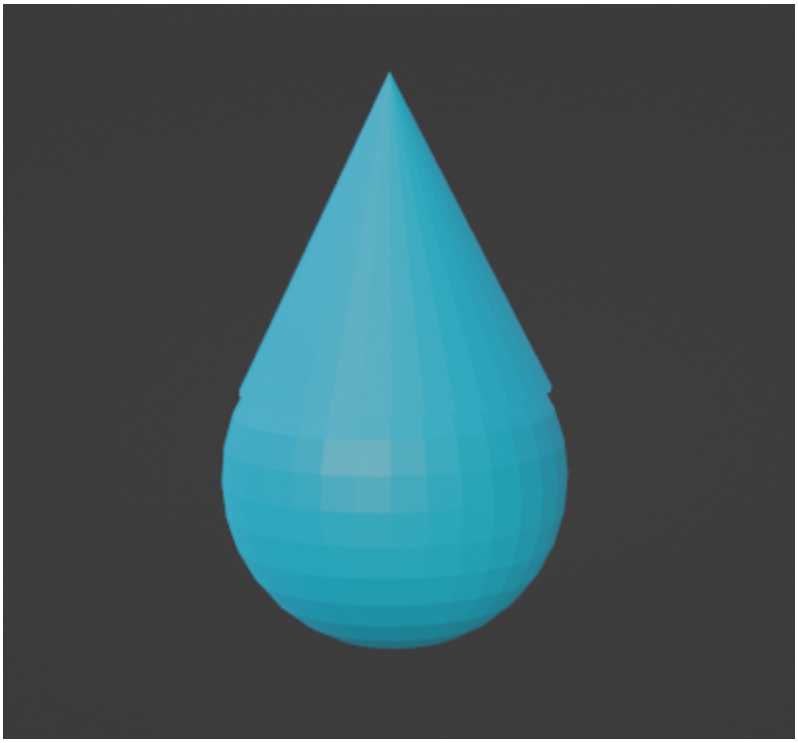
Coeficientul de ceata este calculat cu ajutorul unei formule exponentiale patratice:

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

Pentru efectul propriu-zis de ceata se interpoleaza culoarea cetii cu cea a fragmentului in functie de distanta.

2. Ploaia

Pentru ploaie am modelat o picatura in Blender, pe care mai apoi am adaugat-o in proiect ca si obiect separat.



Pentru a face efectul de ploaie am randat mai multe picaturi la valori aleatoare, atribuindu-le de asemenea si o viteza fiecăreia astefel incat sa cada uniform. Cand o picatura depaseste scena vizibila, ea este din nou randata la o pozitie aleatoare.

3. Transparenta

Transparenta obiectelor este influentata de canalul alfa din culoare. In functie de acest parametru, obiectele primesc o culoare care se combina mai mult sau mai putin cu culoarea obiectelor din spatele lor (culoarea actuala din framebuffer). Daca alfa este 1.0 atunci obiectul este opac, iar daca este 0.0 obiectul este transparent.

4. Fragment discarding

Acest efect este asemanator cu transparenta in sensul ca foloseste acelasi parametru alfa al culorii. Daca acest alfa este sub un anumit prag, fragmentele respective nu se vor mai desena.

5. Lumina punctiforma

Lumina punctiforma este un tip de lumina care are lumineaza radial si ale carei raze isi pierd intensitatea in functie de distanta.
Pentru calculul acesteia am folosit modelul Blinn-Phong care utilizeaza semi-vectorul de reflexie.

6. Lumina directionala

Lumina directionala este un tip de lumina are carei raze sunt paralele si lumineaza la aceeasi intensitate indiferent de distanta.
Pentru calculul acesteia am folosit modelul Phong.

7. Generarea umbrelor

Umbrele au fost generate folosind algoritmul de Shadow Mapping.
Prima etapa al acestui algoritm este rasterizarea scenei din punctul de vedere al luminii de unde luam valorile de adancime si le stocam intr-o harta de adancimi.
A doua etapa e reprezentata de rasterizarea scenei din punctul de vedere al pozitiei camerei si se compara adancimile obiectelor cu cele din buffer-ul de adancimi.
Fragmentele cu o adancime mai mare decat ceea ce a fost stocat anterior in harta de adancime se vor afla in umbra.

Modelul grafic

Scena a fost realizata in aplicatia Blender, cu ajutorul tutorialelor. Obiectele au fost luate de pe diferite site-uri de modele 3D impreuna cu texturile lor. Pentru obiectele care nu aveau texturi, am incercat sa imi creez singura folosid diferite tipuri de imagini de pe internet.

Modelarea anumitor componente (de exemplu picaturile de ploaie) a fost facuta integral de mine, tot in Blender si salvate ca obiecte separate.

Structuri de date

Singura structura de date pe care am implementat-o a fost cea pentru ploaie. Am creat un obiect in Blender reprezentand o picatura de ploaie si o structura care retine pozitia si viteza pentru fiecare entitate in parte.

```
// Struct that represents each raindrop with its position and speed
struct rain {
    glm::vec3 position;
    glm::vec3 speed;
};

// All raindrops
std::vector<rain> rainDrops;
```

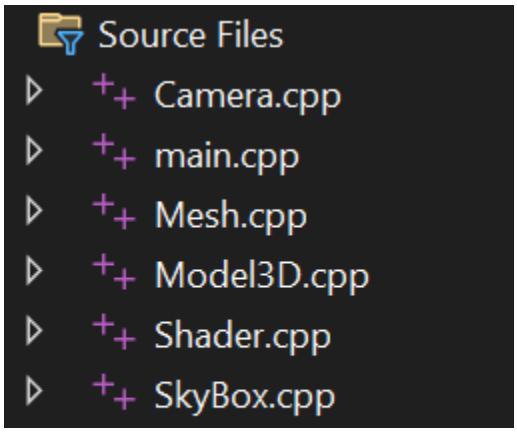
Pentru celelalte functionalitati ale proiectului nu am avut nevoie sa creez structuri noi deoarece cele predefinite din codul din laborator au fost suficiente.

Ierarhia de clase

Pentru acest proiect am folosit un template de la laboratorul 10 de pe Moodle impreuna cu resursele sale.

Clasele din acest proiect:

- Camera : se ocupa de miscarea camerei si schimbarea perspectivei
- Mesh : folosita pentru a defini poligoanele ce construiesc obiectele
- Mode3D : definesc obiectele 3D compuse din mesh-uri
- SkyBox : defineste skybox-ul
- Shader : gestioneaza shader-ele, randeaza obiectele
- Main : implementeaza programul propriu-zis



Prezentarea interfetei grafice utilizator / Manual de utilizare

Utilizatorul poate interactiona in mod direct cu scena prin intermediul:

- Mouse-ului : miscarea acestuia schimba directia de vizualizare a camerei
- Tastaturii : exista un set de taste speciale pentru functionalitati:
    - ☐ W, A, S, D - miscarea camerei pe directiile inainte, inapoi, stanga si dreapta
    - ☐ Q, E - rotirea camerei in directiile stanga si dreapta
    - ☐ J, L - rotirea luminii directionale in directiile stanga si dreapta (implicit declanseaza miscarea umbrelor)
    - ☐ X - vizualizare scena in mod wireframe
    - ☐ Y - vizualizare scena in mod poligonal
    - ☐ Z - vizualizare scena in mod solid
    - ☐ O - pornire/oprire lumina directionala
    - ☐ P - pornire/oprire lmina punctiforma
    - ☐ B - afisarea coordonatelor curente ale camerei
    - ☐ U - tur automat al scenei
    - ☐ F - pornire/oprire ceata
    - ☐ T - pornire/oprire transparenta
    - ☐ M - afisare harta de adancime

Concluzii si dezvoltari ulterioare

Acest proiect a reprezentat o oportunitate de aplicare a conceptelor invatate pe parcursul acestui semestru la laboratoarele de Proiectare Grafica.

Desi scena nu este una foarte complexa, am incercat sa exemplific cat mai multe efecte si functionalitati prezentate la laborator.

Ca dezvoltari ulterioare s-ar putea adauga cateva elemente pentru a creste realismul scenei, cum ar fi:

- adaugarea de sunte specifice (de la pasari, o melodie de fundal)
- adaugarea mai multor obiecte in scena
- adaugarea de animatii
- imbunatatirea turului de vizualizare
- imbunatatirea umbrelor

Referinte

Obiecte 3D:

1 .https://sketchfab.com/feed
2. https://www.cgtrader.com/
3. https://free3d.com/3d-models/
4. https://open3dmodel.com/

Texturi pentru obiecte:

https://www.freepik.com/

Resurse pentru cod:

1. Laboratoare prelucrare grafica: https://moodle.cs.utcluj.ro/course/view.php?id=671
2. Manual functii OpenGL: https://learn.microsoft.com/en-us/windows/win32/opengl/gl-functions

Tutoriale Blender:

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