Proiect OpenGL – Bucatarie în aer liber

Titianu Maria

Ianuarie 2023

Cuprins

[1. Prezentarea temei 3](#_Toc124854018)

[2. Scena și funcționalitățile acesteia 3](#_Toc124854019)

[a. Descrierea scenei și a obiectelor 3](#_Toc124854020)

[b. Funcționalități 3](#_Toc124854021)

[3. Implementare 4](#_Toc124854022)

[a. Algoritmi utilizați 4](#_Toc124854023)

[b. Modelul grafic 4](#_Toc124854024)

[c. Structuri de date 4](#_Toc124854025)

[d. Ierarhia de clase 4](#_Toc124854026)

[4. Prezentarea interfeței grafice / manual de utilizare 4](#_Toc124854027)

[5. Concluzii și dezvoltări ulterioare 5](#_Toc124854028)

[6. Bibliografie 5](#_Toc124854029)

# Prezentarea temei

Scopul acestui proiect a fost realizarea unei aplicații OpenGL care reprezintă o scenă de joc sau film cu animații, prin care să putem naviga. Scena aleasă de mine este o scenă dintr-un joc de gătit, care conține un set de case, între munți, unde este loc de grătar și de o masă. Această scenă are multiple efecte speciale și animații, pe care le voi prezenta ulterior.

# Scena și funcționalitățile acesteia

## Descrierea scenei și a obiectelor

În această scenă, putem observa un sat, împrejmuit de 5 munții. Pe ulița satului, se afla zona de gătit cu grill-ul și masa cu alimente și masa unde te poți așeza mâncarea făcută și o trambulină pentru copii din sat să se distreze. Pe masa se află 3 sticle transparente cu apa și 3 doze de suc reflectorizante.

## Funcționalități

În această aplicație s-au implementat următoarele funcționalități:

* Prin scenă se poate naviga folosind tastatura și mouse-ul
* Se pot observa efectele vizuale speciale (umbre, reflexii, transparență)
* Se poate observa efectul de ceață prin apăsarea unei taste
* Se poate controla starea camerei de la tastatură, aceasta având câteva moduri de vizualizare (liberă, prezentare)
* Se pot porni/opri luminile felinarelor și a ferestrelor de la tastatură
* Se poate trece între modul de noapte și cel de zi de la tastatură
* Se poate vizualiza scena în modurile solid, wireframe, point, flat și smooth
* Se poate roti sursa direcțională de lumină

# Implementare

## Algoritmi utilizați

* Shadow mapping: această funcționalitate are o implementare relativ complexă, pe care o putem împărți în următorii pași:
  + Inițializăm o textură în care vom ține informațiile despre adâncimea fragmentelor
  + Desenăm obiectele din perspectiva luminii folosind proiecția perspectivă corectă pentru tipul de lumină ales (ortografică pentru lumina direcțională și perspectivă pentru punctiformă sau spot) și reținem informațiile despre adâncimea fragmentelor în textura creată.
  + Desenăm scena, verificând pentru fiecare fragment daca acesta se află în spațiul în care trasăm umbrele și daca acesta e umbrit, prin verificarea adâncimii la care se află fragmentul din perspectiva umbrei. Astfel, daca fragmentul este la o adâncime mai mare decât cea din textura de adâncime acesta se consideră umbrit și se desenează mai întunecat.
* Reflexii și skybox: această funcționalitate a fost implementată prin două shadere separate: un shader simplist, care doar făcea sample la un samplerCube pentru a afișa skybox-ul și un shader specializat, care calcula reflecția de pe skybox folosindu-se de poziția fragmentului în spațiul de vizualizare și de normala acestuia, și calcula un sample din vectorul reflectat pe skybox.
* Lumini spot: pentru a implementa luminile mașinii, am adus o extensie la algoritmul de lumini punctiforme. Pentru a implementa funcționalitatea de spot, fiecare lumină are atașată pe lângă parametrii obișnuiți (poziție, culoare, intensitate) noi parametrii, care reprezintă direcția spotului de lumină și unghiul maxim de deschidere (sau „FOV”-ul luminii spot). Pentru a simula funcționalitatea de spot, calculam unghiul dintre direcția luminii și direcția fragmentului, și dacă acesta trecea peste unghiul de FOV nu mai iluminam fragmentul cu acea lumină.

## Modelul grafic

Pentru a realiza modelul grafic am folosit în principal Blender, unde am creat toată scena pe care mai apoi am importat-o în OpenGL. Pentru a păstra separat geamurile de la casă sau geamurile de la felinare și celelalte părțile componente ale scenei si datele despre transparența texturilor, a obiectelor sau despre reflexiile totale ale acestora, am exportat mai multe obiecte (scena\_completa.obj, geamuri.obj, sticla1.obj, doza.obj...etc) pentru a putea afișa aceste propietați ușor în cadrul programului. De asemenea, pentru realizarea scenei am descărcat modele 3D gratuite de pe website-uri cum ar fi Sketchfab [1] sau TurboSquid [2].

## Structuri de date

Ca structuri de date, am folosit în mod extensiv vectorii (vec2, vec3 si vec4) si matricele (mat3, mat4), acestea fiind modalitatea prin care realizam toate transformările din scena. De asemenea, pentru realizarea animației de prezentare, am folosit structura de date vector pentru ușurința cu care putem adăuga noi poziții prin care să treacă camera în timpul rulării.

## Ierarhia de clase

Pentru acest proiect, pentru a reprezenta fiecare poziție prin care trebuie să treacă camera am creat o clasă Keyframe în namespace-ul gps. Aceasta reține informații despre poziția camerei (poziție, țintă, up), momentul de timp la care trebuie să ajungă la acea poziție și modul de interpolare (liniar sau unghiular). De asemenea, clasa oferă o metodă care primește momentul de timp curent și două poziții prin care trebuie să treacă camera, și returnează poziția interpolată. Acest rezultat este unul foarte folositor, fiindcă ne permite realizarea ușoară a animațiilor de prezentare.

În afară de această nouă clasă, am păstrat structura de la laborator, completată și îmbunătățită. Astfel, Camera reprezintă camera și posibilitățile ei de mișcare, iar shaderele, pentru a permite afișarea ușoară pe ecran.

# Prezentarea interfeței grafice / manual de utilizare

După rularea aplicației, ar trebui să apară următoarea scenă pe ecran (Figura 1):



Figură 1 Scena completă pe timpul nopții

După ce scena s-a încărcat și a apărut prima imagine, cu mouse-ul putem mișca camera în modul liber. De la tastatură se pot realiza următoarele acțiuni:

* Schimbarea modului camerei:
  + 1: modul „urmează mașina”
  + 2: modul „în mașină”
  + 9: modul liber
  + 0: modul prezentare
* Controlul camerei:
  + WASD pentru mișcările în plan
  + „Shift stânga” / „Control stânga” pentru mișcările sus-jos
* Controlul mașinii:
  + săgețile sus-jos și stânga dreapta pentru deplasarea mașinii
  + L: pornește / oprește farurile
* Modurile de afișare:
  + F1: solid
  + F2: wireframe
  + F3: point
  + F4: flat
  + F5: polygonal
  + F6: smooth
* R: rotația sursei direcționale
* F: pornește / oprește ceața
* E: zi / noapte
* Q: afișează în consolă poziția curentă a camerei
* L: lumina felinare
* K: lumina geamuri
* H: animația pentru a gătit

# Concluzii și dezvoltări ulterioare

Acest proiect a fost foarte interesant, deoarece m-a ajutat să învăț mult mai multe despre OpenGL și despre modelarea în Blender, și mi-a arătat cât de frumos e acest domeniu al programării.

Ca dezvoltări ulterioare, această scenă ar putea să devină un joc complet, fiind nevoie de implementarea coliziunilor cu pereții, de a putea prăjii carnea și schimbarea texturi pentru un mai mare realism. De asemenea, ar putea să permită utilizatorului o mai mare gama de produse de gătit și o gamă mai largă de ustensile și posibilitatea de a lua obiectele în mană/ dezasambla sandvișul.

# Bibliografie

|  |  |
| --- | --- |
| [1] | I. Sketchfab, „Sketchfab,” 2023. [Interactiv]. Available: https://sketchfab.com/feed. |
| [2] | TurboSquid, „TurboSquid,” 2023. [Interactiv]. Available: https://www.turbosquid.com/. |