*Proiect*

*Prelucrare Grafică*

Iftene Ioan-Florin

Grupa 30231

Profesor coordonator: An academic:

Nandra Constantin 2022-2023

Cuprins

[1. Prezentarea temei 3](#_Toc124698671)

[2. Scenariul 3](#_Toc124698672)

[2.1. Descrierea scenei și a obiectelor 3](#_Toc124698673)

[2.2. Funcționalități 4](#_Toc124698674)

[3. Detalii de implementare 4](#_Toc124698675)

[3.1. Funcții și algoritmi 4](#_Toc124698676)

[3.2. Modelul grafic 10](#_Toc124698677)

[3.3. Structuri de date 10](#_Toc124698678)

[4. Prezentarea interfeței grafice utilizator. Manual de utilizare 10](#_Toc124698679)

[5. Concluzii și dezvoltări ulterioare 11](#_Toc124698680)

[6. Referințe 11](#_Toc124698681)

# Prezentarea temei

Obiectivul acestui proiect este realizarea unei prezentări fotorealiste a unor scene de obiecte 3D utilizând librăriile OpenGl, GLFW, GLM. Utilizatorul trebuie să aibă posibilitatea de a naviga în scenă prin intermediul tastaturii și a mouse-ului. Scena trebuie să poată fi vizuzlizată în mai multe moduri (solid, wireframe, punctiform) și să conțină diferite surse de lumină, obiecte texturate, umbre, animații ale unor elemente și diferite efecte, precum ceață, obiecte transparente, reflexii.

# Scenariul

## Descrierea scenei și a obiectelor

Scena prezintă un peisaj dintr-o zonă muntoasă, în mijlocul căreia se află un loc cu pietriș care permite crearea unor construcții. Prin urmare, se vor putea observa mai multe elemente în această zonă. Într-un colț al scenei se află o navă de timp Imperial Fighter din StarWars, prăbușită. În imediata vecinătate a ei se află roboții BB8, C3PO, care are o vopsea ce reflectă lumina și R2D2, care are deasupra capului un bec transparent ce se aprinde când are o idee. Cei 3 stau față în față cu un grup format din 3 lupi, un urs și baby Yoda, care poartă ochelari de soare ce reflectă skybox-ul.

Mergând mai departe în scenă, spre celălalt capăt al locului cu pietriș din munți, pe o parte se află o navă spațială, al cărei geam este transparent, iar pe cealaltă parte, camuflat în munți, se află un robot. Robotul este plasat pe un stand reflexiv și are capacitatea de a împușca cu lasere, reprezentate de obiecte cu formă cilindrică.

În celălalt capăt al locului, într-un colț se află un turn de observare, format din lemn, iar în cealălalt se găsește un portal, în spatele căruia este o grămadă de cranii. Dacă intrăm în portal cât acesta este activat, ne vom ”teleporta” într-o zonă din munți în care se află un lac. În acel lac se află o creatură marină, pe care o putem observa în întregime datorită transparenței apei.

Toată această scenă a fost adăugată într-un cub texturat, numit skybox. Textura aplicată pe skybox descrie o scenă dintr-o zonă montană, similară cu cea realizată în cadrul proiectului, contribuind astfel la foto-realismul scenei.

## Funcționalități

* Utilizatorul are posibilitatea de a naviga prin scenă atât cu ajutorul tastaturii, cu tastele WASD, cât și cu ajutorul mousului. Pentru a naviga doar cu mouse-ul, utilizatorul trebuie să îl țină orientat spre direcția în care dorește să meargă și să dea scroll pentru a se apropia sau pentru a se îndepărta, în funcție de ce dorește să facă. Există și opțiunea de a lăsa aplicația să prezinte scena de una singură, apăsând tasta 8.
* Legat de sursele de lumină, utilizatorul are posibilitatea de a roti sursa de lumină direcțională și de a o activa/dezactiva pe cea de tip spot.
* Scena poate fi vizualizată în modurile solid, wireframe și punctiform
* Portalul se activează prin apăsarea unei taste. Utilizatorul știe că acesta este activat datorită animației de rotire aplicată portalului. Când utilizatorul intră în portal și acesta este activat, va apărea efectul de teleportare și el va fi trimis într-o altă zonă a scenei, în care se află un lac transparent, prin care se poate vedea un monstru marin.
* Asupra scenei se poate aplica efectul de ceață prin apăsarea unui buton.
* Robotul de pe cubul reflexiv din munte poate împușca cu lasere turnul din lemn. La apăsarea unei taste, acesta începe să emită laserul înspre turn. Dacă se ține suficient de mult apăsat pentru a ține laserul în funcțiune, turnul se va distruge și se va transforma într-o grămadă de materiale.
* Lumina direcțională cauzează crearea umbrelor pentru obiectele care se află pe direcția sursei de lumină. Umbrele se rotesc odată cu sursa de lumină și se modifică în funcție de poziția acesteia.

# Detalii de implementare

## Funcții și algoritmi

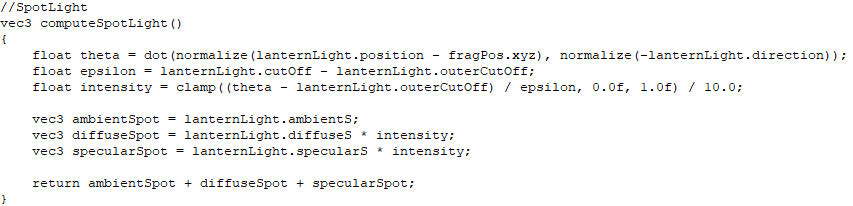
**Camera [1]**

În fișierul Camera.cpp am implementat funcțiile getViewMatrix(), move() și rotate(), care sunt folosite pentru a permite utilizatorului să se miște în 4 direcții: înainte, înapoi, la stânga și la dreapta, dar și să se rotească cu mouse-ul.

În fișierul main.cpp am implementat funcțiile mouseCallback() și scrollCallback() pentru a facilita vizualizarea scenei cu ajutorul mouse-ului.

**Lumini [2][5]**

În acest proiect au fost folosite trei tipuri de surse de lumină. O lumină este de tip direcțional, una de tip spot și una de tip punctiform. Pentru lumina de tip direcțional au fost urmate instrucțiunile din îndrumătorul de laborator de la Prelucrare Grafică.

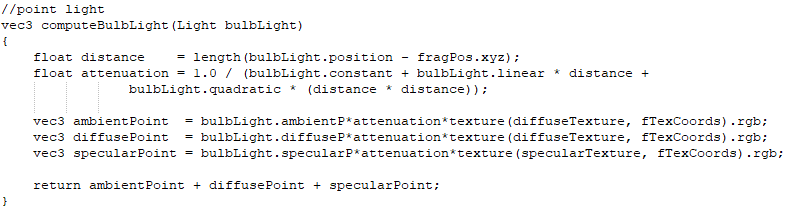
Lumina de tip spot a fost creată utilizând instrucțiunile de pe learnopengl.com. Codul pentru această sursă de lumină a fost scris în principal în fragment shader-ul shaderStart.frag. Am folosit o structură de date denumită SpotLight, care conține toate atributele necesare pentru lumina de tip spot. Am declarat apoi o variabilă uniform, lanternLight, de tip SpotLight, ale cărei atribute vor fi timise din aplicație, prin intermediul funcțiilor initSpotLight\_On(), respectiv initSpotLight\_Off(), din fișierul main.cpp. Revenind înapoi în shader, în funcția computeSpotLight() am folosit atributele primite prin intermediul variabilei uniform pentru a construi lumina de tip spot. Parametrii theta și epsilon au fost inițializați conform instrucțiunilor, ținând cont de poziție și direcție pentru theta, respectiv de cutOff și outerCutOff pentru epsilon, iar intensitatea a fost calculată folosind cei doi parametrii, împreună cu unghiurile cutOff și outerCutOff. Funcția returnează suma dintre componentele ambient, diffuse și specular, primite de la variabila uniform și înmulțite cu intensitatea calculată.

Figură 1: Funcția computeSpotLight

În funcția main din shader, am adăugat valoarea returnată de computeSpotLight la variabila color, pentru a face vizibilă lumina de tip spot.

Rezultatul obținut pentru lumina de tip spot este următorul:

Figură 2: Sursă de lumină de tip spot

Lumina de tip punctiform a fost implementată utilizând instrucțiunile de pe același site ca și în cazul luminii de tip spot. Am folosit o structură denumită Light care conține toate detaliile necesare pentru acest tip de lumină și am utilizat o variabilă uniformă de tipul structurii în funcția computeBulbLight din fragment shader-ul shaderStart pentru a crea lumina. Componentele ambient, difuz și specular ale culorii luminii punctiforme au fost înmulțite cu atenuarea, parametru influențat de poziția luminii, distanță și cei 3 factori, constant, linear și quadratic, care influențează cât de departe bate lumina.

Figură 3: Sursă de lumină de tip punctiform

Figură 4: Funcția computeBulbLight

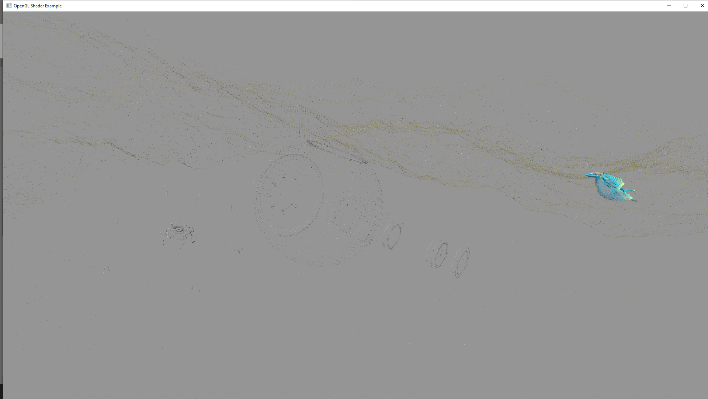
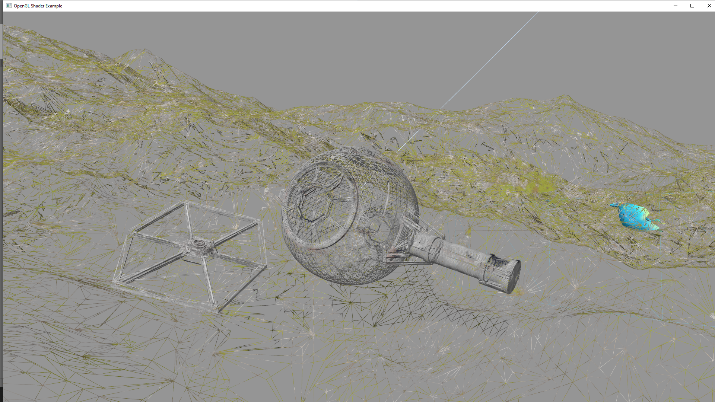
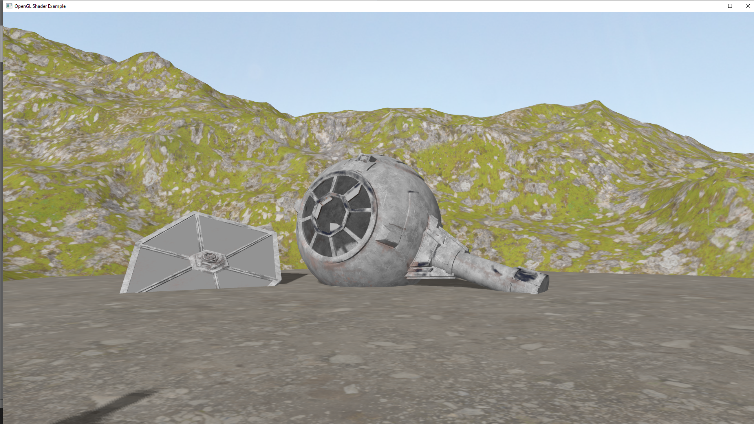
**Moduri de vizualizare ale scenei [3]**

În acest proiect, scena poate fi vizualizată în trei moduri diferite: solid, wireframe și punctiform. Pentru a implementa aceste moduri, a fost folosită funcția glPolygonMode, cu primul parametru fiing GL\_FRONT\_AND\_BACK și al doilea depinzând de modul dorit.

Pentru vizualizarea wireframe, al doilea parametru al funcției este GL\_LINE. În acest mod de vizualizare, muchiile vecine ale poligoanelor sunt desenate ca și segmente de linie.

Pentru vizualizarea punctiformă, al doilea parametru este GL\_POINT. În acest mod de vizualizare, sunt desenate doar punctele care marchează începutul unei noi muchii a unui poligon.

Modul de vizualizare solid este cel cu care suntem obișnuiți, în care interiorul poligoanelor este plin, fiind desenate toate punctele și muchiile din care sunt alcătuite. GL\_POINT este modul care trebuie specificat funcției glPolygonMode pentru acest tip de vizualizare.

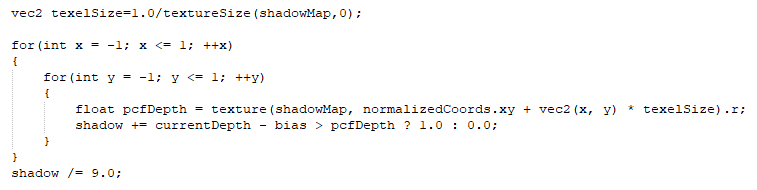


Figură 5: Modurile de vizualizare wireframe (stanga, sus), punctiform (dreapta, sus) și solid (jos)

**Generarea umbrelor[5]**

Umbrele au fost adăugate în proiect folosind tehnica Shadow Mapping, descrisă în îndrumătorul de laborator. Shadow mapping este o tehnică multi-trecere care utilizează texturi de adâncime pentru a decide dacă un punct se află în umbră sau nu. Cheia este aceea de a observa scena din punctul de vedere al sursei de lumină. . Orice parte a scenei care nu este direct observabilă din perspectiva luminii va fi în umbră.

Algoritmul rasterizează mai întâi scena din punctul de vedere al luminii, singurele informații relevante în această fază fiind valorile de adâncime ale punctelor. Aceste valori sunt stocate într-o hartă de umbră (de adâncime), care va fi umplută cu valorile relevante ale adâncimii, din punctul de vedere al sursei de lumină. După acest pas, scena este rasterizată din punctul de vedere al observatorului. Se compară adâncimea fiecărui fragment vizibil cu valorile de adâncime din harta umbrelor. Fragmentele care au o adâncime mai mare decât cea care a fost stocată anterior în harta de adâncime nu sunt direct vizibile din punctul de vedere al luminii și sunt, prin urmare, în umbră.

Problemele legate de shadow acne și de supra-eșantionare au fost rezolvate conform instrucțiunilor din lucrarea de laborator. Pentru ca umbrele să nu mai pară atât de pixelate, am folosit algoritmul PCF[4]. Acest algoritm ia în considerare și zonele adiacente unui punct atunci când se verifică valorile din harta de umbre și face o medie pe care o va folosi atunci când decide ce valoare are umbra în acel loc.

Figură 6: Algoritmul PCF (Percentage-Closer Filtering)

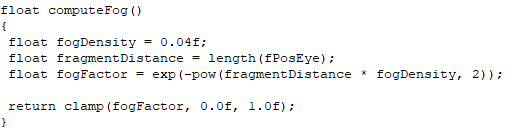
**Animația portalului**

Pentru a activa portalul, utilizatorul apasă pe un buton, moment în care acesta începe să se învârtă. Acest efect vizual a fost obținut prin rotirea obiectului care reprezintă portalul în jurul centrului său.

**Animația laserului**

Laserul cu care ”împușcă” robotul este reprezentat de 2 cilindrii. Când utilizatorul apasă pe buton pentru a porni laserul, acesta se scalează cu 0.1 în plus în fiecare moment în care butonul este apăsat, oferiind astfel efectul vizual de laser.

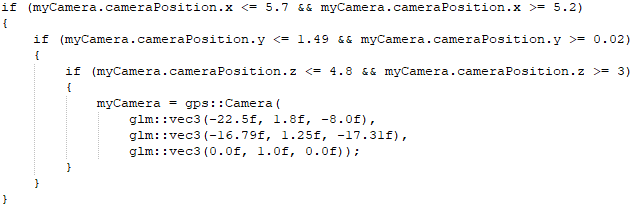
**Efectul de ceață[5]**

Utilizatorul are opțiunea să aplice și ceață asupra scenei. Algoritmul care implementează ceața a fost scris în funcția computeFog din shader-ul fragment shaderStart. Când ceața se dorește a fi activată, culoarea pe care o returnează shader-ul este un mix între culoarea calculată până în acel punct, culoarea ceții și factorul de ceață, returnat de funcția computeShadow, care ține cont de distanța la care se află observatorul și densitatea ceții.

Figură 7:Funcția computeFog

**Efectul de teleportare**

Când utilizatorul intră în portal în momentul în care acesta este activat, el va fi mutat într-un alt punct al scenei, în care va avea parte de o altă priveliște față de cea din zona din care s-a ”teleportat”. Acest efect a fost obținut luând în calcul punctele de extrem ale obiectului ce reprezintă portalul pe toate cele 3 axe și modificând cameraPosition și cameraDirection în momentul în care utilizatorul pătrunde în portal (cameraPosition este între punctele de extrem).



Figură 8: Schimbarea camerei în momentul intrării în portal

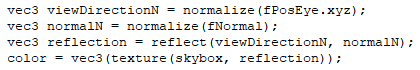
**Distrugerea turnului**

În momentul în care laserul intră în contact cu turnul, dacă butonul de pe care e controlat este ținut apăsat îndeajuns de mult, turnul se va distruge. Acest efect a fost obținut prin verificarea coordonatelor la care a ajuns capătul laserului, folosind gradul de scalare și prin înlocuirea turnului cu un obiect care să indice utilizatorului că turnul a fost distrus de către laser.

**Obiecte transparente**

În scenă se află trei obiecte transparente: geamul navei spațiale, lacul și becul de deasupra lui R2D2. Pentru a obține transparența, aceste obiecte sunt desenate ultimele, iar înainte de a le desena a fost activat GL\_BLEND, pentru a putea ține cont și de factorul de transparență, care este ultimul parametru al vectorului de 4 elemente ce reprezintă culoarea în fragment shader.

**Obiecte reflectorizante**

Robotul C3PO, cubul pe care se află robotul din munți care împușcă cu laser și lentilele ochelarilor lui baby Yoda sunt obiecte ce reflectă skybox-ul. Pentru a obține acest efect de reflexie am folosit un alt shader, numit shaderC3 în care culoarea este calculată ținând cont de direcția de vizualizare, textura skybox-ului, poziția observatorului și folosind funcția reflect(), astfel obținându-se efectul de reflexie.

Figură 9: Obținerea culorii pentru obiectele ce reflectă skybox-ul

**Prezentare automată a scenei**

Utilizatorul are opțiunea de a apăsa un buton și a lăsa aplicația să mute camera prin scenă astfel încât el să vadă toate elementele din care este ealcătuită scena. Acest lucru a fost obținut scriind într-un fișier poziția camerei și direcția în care privește, în fiecare moment în timpul unei rulări a aplicației în care a fost vizitată întraga scenă. La apăsarea tasatei 8, se citesc acele date din fișier și se atribuie parametrilor camerei, astfel obținându-se o prezentare automată a scenei, în care utilizatorul nu trebuie să mai folosească mouse-ul sau tastatura pentru a vedea întreaga scenă.

## Modelul grafic

Obiectele prezente în scenă au fost descărcate de pe diferite site-uri[6][7], în format .obj sau .fbx și apoi importate în Blender. Texturile obiectelor sunt imagini în format .jpg sau .png, descărcate împreună cu obiectele sau create în Paint, pentru obiectele care nu aveau texturi. Toate obiectele au fost aranjate în Blender și poziționate în locul dorit, iar apoi au fost exportate în format .obj și încărcate în proiect. Obiectele pentru care au fost făcute animații (portalul, laserele), cele care au proprietăți speciale, precum reflexii sau transparență, sau cele care urmează a fi înlocuite (turnul de lemn) au fost exportate și introduse în proiect ca și obiecte individuale. Restul obiectelor din scenă au fost exportate și introduse ca și un singur obiect .obj.

## Structuri de date

Pentru a realiza proiectul, pe lângă structurile de date care aparțin de biblioteca OpenGL, am creat o structură de date, numită SpotLight, care să ajute la introducerea unei lumini de tip spot în proiect. Această structură conține câmpurile position, direction, cutOff, outerCutOff, ambient, diffuse și specular. Structura se află în fragment shader-ul shaderStart și este folosită pentru a declara o variabilă de tip uniform și a construi lumina de tip spot în funcția computeSpotLight, după cum a fost explicat în secțiunea 4.1.

O altă structură de date folosită este Light, care a fost folosită tot pentru crearea unui alt tip de sursă de lumină, de această dată fiind vorba de lumina punctiformă. În această structură se rețin date despre poziția sursei, culoare (componentele ambient, difuz și specular) și despre coeficienții care influențează raza de acțiune a luminii: constant, linear și quadratic.

# Prezentarea interfeței grafice utilizator. Manual de utilizare

În momentul în care aplicația este lansată, scena va apărea pe ecranul utilizatorului. Cursorul este dezactivat pentru o mai bună interacționare cu aplicația și pentru o libertate mai mare de mișcare în scenă. Pentru a naviga prin scenă, se poate folosi doar mouse-ul, utilizând rotița de scroll pentru apropierea/ îndepărtarea de un obiect și ținându-l orientat spre direcția dorită de deplasare, sau se pot folosi tastele pentru mișcare față, spate, stânga, dreapta, împreună cu orientarea mouse-ului spre direcția dorită.

Pentru a folosi diferite funcționalități ale aplicației, se utilizează următoarele taste:

1. Mișcare înainte: **W**
2. Mișcare la stânga: **A**
3. Mișcare în spate: **S**
4. Mișcare la dreapta: **D**
5. Pornire/Oprire prezentare automată a scenei: **8**/**9**
6. Schimbarea direcției luminii direcționale: **J**/**L**
7. Activare/Dezactivare lumină de tip spot: **X/C**
8. Activare/Dezactivare lumină de tip punctiform (becul): **I**/**O**
9. Activare/Dezactivare ceață: **F**/**G**
10. Vizualizare scenă în mod Wireframe: **1** (unu)
11. Vizualizare scenă în mod solid: **2**
12. Vizualizare scenă în mod punctiform: **3**
13. Activare portal: **P**
14. Activare laser: **Z**
15. Închidere aplicație: **Esc**

# Concluzii și dezvoltări ulterioare

Am reușit să realizez toate cerințele pe care mi le-am propus în acest proiect și m-am familiarizat mai bine cu biblioteca OpenGL și cu progrmaul de modelare Blender. Am învățat cum pot să îmbunătățesc umbrele obiectelor, cum pot face obiecte care să reflecte mediul sau care să fie transparente și am reușit să creez animații utilizând translații, rotații și scalări. Am dat utilizatorului posibilitatea de a se mișca în scenă atât cu mouse-ul, cât și cu tastatura, opțiunea de a putea mișca sursa de lumină și de a vizualiza scena în mai multe moduri. Proiectul ar putea fi dezvoltat prin introducerea mai multor obiecte în scenă, care să poată fi mișcate în diferite direcții; prin opțiunea de a mișca robotul care împușcă cu lasere și de a putea selecta și alte ținte; prin adăugarea altor tipuri de surse de lumină sau prin animarea creaturii marine și a lacului.

# Referințe

*[1] Camera*. (fără an). Preluat de pe https://learnopengl.com/Getting-started/Camera

*[2] Light casters*. (fără an). Preluat de pe https://learnopengl.com/Lighting/Light-casters

*[3] glPolygonMode*. (fără an). Preluat de pe https://registry.khronos.org/OpenGL-Refpages/gl4/html/glPolygonMode.xhtml

*[4] Shadow Mapping*. (fără an). Preluat de pe https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

*[5] Îndrumător de laborator*. (fără an). Preluat de pe moodle.cs.utcluj.ro

*[6] CGTrader*. (fără an). Preluat de pe https://www.cgtrader.com

*[7] Free3D*. (fără an). Preluat de pe https://free3d.com