Logo

Description automatically generated

Proiect Prelucrare Grafică

Oraș pe timp de seară

Pașcan Daniela Maria

**Cuprins**

1. Prezentarea temei........................................................................................3
2. Scenariu.......................................................................................................3
   1. Descrierea scenei și a obiectelor...........................................................4
   2. Funcționalități.......................................................................................5
3. Detalii de implementare..............................................................................6
   1. Funcții și algoritmi................................................................................6

3.2.1 Motivarea abordării alese.............................................................8

* 1. Modelul grafic.......................................................................................8
  2. Structuri de date....................................................................................9
  3. Ierarhia de clase....................................................................................9

1. Prezentarea interfeței grafice.....................................................................10
2. Manual de utilizare....................................................................................13
3. Concluzii...................................................................................................13
4. Dezvoltări ulterioare..................................................................................14
5. Referințe....................................................................................................14
6. **Prezentarea temei**

Pentru acest proiect am optat pentru a face o scenă destul de simplă, atrăgătoare și cât mai plăcută vizual. Am folosit mediul de creare și de editare Blender pentru a produce o scenă, pe care mai apoi am încărcat-o în VisualStudio. În continuare am folosit instrumentele provenite de la biblioteca OpenGL pentru a adăga efecte scenei și pentru a manevra mult mai ușor obiectele pe care am ales să le adaug.

Conceptul acestei scene este un oraș micuț și primitor, peste care se așază ușor noaptea și se poate observa atmosfera primitoare a parcului de copii cu felinare și a unei ceți ușoare.

Ca și scop al acestui proiect am avut folosirea bibliotecilor OpenGL, GLFW, GLM, ce au fost prezentate pe parcursul laboratoarelor.

1. **Scenariu**

A video game of a neighborhood

Description automatically generated

Scena a fost creată în Blender, pentru a fi mult mai ușor de importat în proiect, și de asemenea de modificat.

Am început prin a construi terenul pe care am dorit să am orașul, folosind instrumentele de sculptare din Blender, iar mai apoi am adăugat pe rând fiecare obiect, am pus textura potrivită pe el iar mai apoi i-am fixat locul în scenă.

* 1. **Descrierea scenei și a obiectelor**

Scena creată în Blender conține:

* O casă mare și modernă
* O piscină, aparținând locuitorilor casei, un gnom și un balansuar
* Un câine, împreună cu căsuța sa
* Două scaune și o umbreluță pentru relaxare în fața piscinei
* O șosea ce străbate întreg orășelul
* Un parc pentru copii ce conține bucăți de pavaj, două băncuțe, un leagăn si un topogan, precum și două felinare ce asigură iluminarea parcului
* O altă casă, ce se poate asemăna cu o fermă
* La fermă sunt puși într-un spațiu îngrădit trei cai, împreună cu mâncarea lor, o moară, o sperietoare, un câmp cultivat
* Pe dealurile scenei se regăsesc câteva pietre mari
* In spatele cailor, cât și în alte părți ale scenei se află copaci
* Un motel, special pentru trecătorii care vor să mai rămână în oraș

La final, am dorit să adaug și o mașină pe șosea, care poate să se miște în față și în spate după cum dorește utilizatorul prin apăsarea unor taste.

De asemenea, o pasăre care se învârte în cerc deasupra scenei și o minge care poate fi rostogolită în față și în spate de către utilizator.

Am dorit ca scena să fie cuprinsă de o atmosferă îmbietoare, cu o lumină caldă și cu peisaj liniștitor. De asemenea, am dorit să incorporez și posibilitatea de a avea ceață în scenă, pentru un aer mai sumbru și interesant.

* 1. **Funcționalități**

La început, aplicația se deschide cu camera poziționată undeva în mijlocul scenei, la nivelul solului pentru a putea privi obiectele dintr-o perspectivă corect dimensionată.

Dacă dorim să ne mișcăm prin scenă, avem posibilitatea de a o face utilizând atât tastatura cât și mouse-ul. Putem să rotim camera atât pe axa Ox cât și pe axa Oy, iar mai apoi putem să ne mișcăm în voie față, spate, stânga, dreapta, tot prin intermediul tastaturii. Dacă dorim să ne folosim și de mouse, putem să reglăm direcția în care privește camera prin intermediul acestuia.

Din punct de vedere al luminii, în scenă am folosit de două tipuri. În primul rând, avem lumină direcțională globală, care poate fi identificată prin pătratul alb pe care l-am lăsat în scenă pentru a putea vedea exact din ce direcție cade lumina pe obiecte. Aceasta este folosită și pentru a crea umbrele obiectelor din scenă și utilizatorul are posibilitatea de a o rotii și de a schimba direcție în care se află umbrele. Am reglat culoarea acestei lumini pentru a da impresia de înserare în oraș.

Un alt tip de lumină folosit este cel punctiform. Am avut nevoie de două astfel de lumini pentru a face felinarele din parc să meargă. Aceste lumini sunt galbene și au intensitatea și raza predefinită în aplicație, deci nu pot fi manevrate de către utilizator.

De asemenea, scena dispune și de efectul de ceață. Aceasta este din nou manevrabilă, utilizatorul poate să regleze nivelul de ceață tot prin intermediul tastaturii. Atunci când aplicația este pornită, nu avem ceață.

Pentru a putea vedea scena în modurile solid, wireframe și texturat ne folosim din nou de tastatură, iar pentru a putea vedea harta de adâncime și modul în care lumina lovește obiectele pentru a genera umbre avem din nou o tastă specială.

La final, mașina din mijlocul scenei poate fi și ea mutată în față și în spate de către utilizator, din nou, prin tastatură.

Aplicația mai conține și o animație ce dorește să prezinte întreaga scenă pentru utilizator, în mod automat, prin apăsarea tastei ENTER. Prin animație se încearcă rotirea și mișcarea prin scenă pentru a observa toate obiectele prezente.

1. **Detalii de implementare**

Deoarece, pentru a obține anumite efecte vizuale în scenă avem nevoie de anumiți algoritmi de prelucrare grafică am ales să utilizez algoritmii și tehnicile învățate pe parcursul laboratoarelor pentru că au fost foarte bine explicate și ușor de implementat.

**3.1 Funcții și algoritmi**

**Modelul Blinn-Phong**

Iluminarea Blinn-Phong este un model de iluminare utilizat în grafica computerizată pentru a simula aspectul suprafețelor iluminate. Acest model este o îmbunătățire a modelului Phong și include un termen suplimentar pentru a lua în considerare strălucirea obiectelor în funcție de direcția privirii camerei.

* Ambient Light :

Reprezintă iluminarea constantă pe toată suprafața obiectului și pe toată scena. Aceasta nu depinde de direcția luminii sau a camerei, în modelul Blinn-Phong, acesta este de obicei asociată cu o culoare ambientală.

* Diffuse Reflection :

Reprezintă iluminarea difuză în funcție de unghiul dintre normala la suprafață și direcția luminii, ea este responsabilă pentru culoarea vizibilă a obiectului în condiții de iluminare difuză.

Se calculează utilizând produsul scalar dintre normala la suprafață și direcția luminii.

* Specular Reflection :

Reprezintă strălucirea suprafeței, care depinde de direcția privirii camerei și de direcția reflectării luminii. În modelul Blinn-Phong, se utilizează un vector intermediar numit vectorul Blinn.

Se calculează utilizând produsul scalar dintre vectorul Blinn și direcția privirii camerei. Este asociat cu un coeficient specular și cu o putere speculară (coeficientul shininess).

* Vectorul Blinn:

Este un vector normalizat calculat între direcția luminii și direcția privirii camerei. Este utilizat pentru a determina strălucirea în funcție de direcția observatorului și direcția iluminării.

Modelul de iluminare Blinn-Phong oferă rezultate vizuale plăcute și este folosit în principal în aplicații de grafică 3D pentru a simula aspectul materialelor sub iluminare. Este o îmbunătățire față de modelul Phong, în special în ceea ce privește strălucirea și aspectul mai natural al suprafețelor iluminate.

**Shadow Mapping**

Shadow mapping este o tehnică comună folosită în grafica 3D pentru a simula umbre realiste produse de iluminarea din surse de lumină. Această tehnică implică generarea unei hărți de adâncime (depth map) care reprezintă distanțele de la sursa de lumină la punctele din scenă. Această hartă de adâncime este apoi utilizată pentru a determina dacă un anumit punct din scenă este iluminat sau umbrit.

Pentru generarea hărții de adâncime se alege o sursă de lumină (de obicei, o lumină direcțională sau o lumină punctiformă) iar apoi o cameră este poziționată în punctul luminii, iar o hartă de adâncime este generată de la perspectiva acestei camere. Această hartă este denumită și "shadow map".

În cadrul acestei hărți de adâncime, valorile reprezintă adâncimile punctelor respective în funcție de perspectiva camerei luminii.

Se utilizează harta de adâncime generată anterior pentru a determina dacă un anumit punct din scenă este iluminat sau umbrit în raport cu sursa de lumină.

Pentru fiecare fragment din scenă, se calculează coordonatele corespunzătoare din harta de adâncime și se compară cu valoarea adâncimii din harta de adâncime.Dacă adâncimea fragmentului este mai mare decât adâncimea stocată în harta de adâncime, fragmentul este în umbră; altfel, este iluminat.

Poate apărea problema numită "shadow acne" dacă nu sunt luate măsuri adecvate. Această problemă poate fi cauzată de erori de precizie în adâncime și poate fi tratată prin adăugarea unui mic bias pentru a evita comparații stricte.

Tot cu ajutorul materialelor din laboratoare am reușit să implementez în aplicația mea și ceața, care este o tehnică specifică cu efect vizual spectaculos.

Proiectul meu conține următoarele funcții principale:

* **makeAnimation()** – această funcție este responsabilă de pornirea și de oprirea animației de prezentare a scenei. Camera este poziționată mai întâi undeva în centrul scenei iar mai apoi se mișcă pentru a prezenta toate. obiectele, de asemenea ne este prezentată și ceața și mașina care se mișcă.
* **processMovement()** – funcția este responsabilă cu specificarea tastelor și a rolului acestora.
* **initSkybox()** – aici se vor procesa imaginile alese pentru a încapsula scena într-un mod cât mai realist cu conținutul acesteia și a face trecerea mai ușoară.
* **initModels()** – în această funcție sunt inițializate obiectele 3D de care avem nevoie.
* **initShaders()** – aici sunt inițializate shadere-le (basic, depthMap, lightCube, skyBox).
* **initUniforms()** – inițializează toate uniforms pentru shader.
* **initFBO()** – a fost folosită pentru harta de adâncime, primul pas al algoritmului de Shadow Mapping fiind rasterizarea din punctul de vedere al luminii, iar mai pe urmă rasterizarea întregii scene.
* **computeLightSpaceTrMatrix()** – aici se va returna matricea de vizualizare a scenei, în funcție de lumină.
* **drawObjects(gps::Shader shader, bool depthPass)** – această funcție desenează obiectele 3D în aplicație, dar ține cont dacă utilizatorul dorește ca acestea să fie desenate normal sau în harta de adâncime.
* **renderScene()** – aici sunt poziționate obiectele în scenă și se face shadow map.

**3.1.1 Motivarea abordării alese**

Am decis să folosesc în proiectul meu algoritmii explicați mai sus deoarece sunt foarte actuali și buni, un plus fiind faptul că au fost înțeleși de pe parcursul laboratoarelor.

**3.2 Modelul grafic**

Obiectele utilizate în proiectul nostru au fost preluate în principal de pe platforme precum Sketchfab, free3D și TurboSquid sub forma de fișiere .obj și .fbx. Aceste obiecte au inclus adesea texturi predefinite. Pentru a le integra corespunzător în proiect, am importat aceste fișiere în mediul de modelare Blender. Aici, am aplicat texturi, le-am poziționat în scena noastră la locul potrivit, le-am combinate sau separat, am reglat scala și am realizat orice modificări necesare. Ulterior, am exportat obiectele în formatul .obj, fiecare fiind însoțit de un fișier .mtl care conține informații detaliate despre texturi.

Fișierele .obj au fost integrate în proiectul nostru, permitându-ne să le desenăm, să le animăm, să le redimensionăm, să le rotim și să le gestionăm în funcție de cerințele noastre specifice.

**3.2 Structuri de date**

Pentru acest proiect am folosit structurile de date din bibliotecile GLM, GLEW, adică vectori, matrici, int-uri, float-uri mai speciale.

O altă structură de date este și clasa Model3D, loc în care se vor reține coordonatele obiectului nostru cu format obj.

**3.4 Ierarhia de clase**

Pentru realizarea acestui proiect, am folosit template-ul special dat de la laborator care conține mai multe clase ajutătoare pentru încărcarea obiectelor 3D în scenă de exemplu sau pentru a face legătura cu shaderele.

Proiectul conține următoarele clase:

* Camera.cpp și Camera.hpp
* Mesh.cpp și Mesh.hpp
* Model3d.cpp și Model3D.hpp
* Shader.cpp și Shader.hpp
* SkyBox.cpp și SkyBox.hpp
* Stb\_image.h și Stb\_image.cpp
* Tiny\_obj\_loader.h și Tiny\_obj\_loader.cpp
* Window.h și Window.cpp

1. **Prezentarea interfeței grafice**

A car driving on a road next to a house

Description automatically generated

A playground with a bench and a bench on a hill

Description automatically generated with medium confidence

A screenshot of a video game

Description automatically generated

**Modul cu ceață**

****

**Modul line**

**A wireframe of a playground

Description automatically generated**

**Modul point**

**A screenshot of a video game

Description automatically generated**

1. **Manual de utilizare**

Utilizatorul poate interacționa cu scena prin intermediul următoarelor butoane de la tastatură:

* A – mișcare cameră în stânga
* S – mișcare cameră în spate
* W – mișcare cameră în față
* D – mișcare cameră în dreapta
* Q – orientare cameră în jos
* E – orientare cameră în sus
* Z – orientare cameră înspre dreapta
* X – orientare cameră înspre stânga
* 1 – modul de vizualizare normal
* 2 – modul de vizualizare line
* 3 – modul de vizualizare point
* 9 – creștere intensitate ceață
* 0 – scădere intensitate ceață
* UP – mișcare mașină în față
* DOWN – mișcare mașină în spate
* ENTER – pornire animație de vizualizare
* BACKSPACE – oprire animație de vizualizare
* J – rotire sursă de lumină stânga
* L – rotire sursă de lumină dreapta
* M – afișare depth map
* Y – mișcare minge în față
* H – mișcare minge în spate

1. **Concluzii**

Prin intermediul acestui proiect am învățat să lucrez atât cu OpenGL, cât și cu Blender pentru a alege obiecte, a le pune o textură potrivită, a le poziționa corect în spațiul scenei și mai apoi a aplica algoritmi și transformări asupra lor. Au fost de mare ajutor materialele din timpul laboratoarelor, deoarece la proiectul final a trebuit doar să combin mai multe tehnici deja implementate și învățate.

Consider că acest proiect a ajutat destul de mult la dezvoltarea mea personală și mi-a îmbunătățit atât răbdarea cât și simțul estetic.

1. **Dezvoltări ulterioare**

Consider că ar fi frumos de adăugat la scena mea mai multe animații cum ar fi mișcarea leagănului în față și în spate, sau mișcarea cailor continuă și posibil mai multe lumini punctiforme pentru o atmosferă și mai primitoare a caselor.

Aș mai vrea, poate pe viitor să mai adaug și ploaie în scenă.

1. **Referințe**

<https://www.turbosquid.com/Search/3D-Models/free?page_num=11>

<https://free3d.com/3d-models/>

<https://learnopengl.com/Lighting/Multiple-lights>