UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Documentație proiect

Prelucrare Grafică

Student: Rotariu Laura-Alexandra
Grupa: 30234

FACULTATEA DE AUTOMATICĂ
ȘI CALCULATOARE

2023-2024

Cuprins

1. Prezentarea temei

Proiectul are ca și scop realizarea unei prezentări fotorealiste a unei scene de obiecte 3D, utilizând librăriile prezentate la laborator (OpenGl, GLFW, GLM). OpenGL este o librărie grafică tridimensională pentru dezvoltarea aplicațiilor grafice interactive, GLFW este o bibliotecă C care furnizează un set de instrumente pentru crearea și gestionarea ferestrelor, a contextelor OpenGl și a evenimentelor de intrare, iar GLM este o bibliotecă C++ pentru matematica graficelor.

Eu am ales să modelez o scenă cu câmp și apă, în care sunt două lumi: lumea reală, cu animale, construcții, utilaje și lumea fantastică, cu ponei, curcubeu imaginar, castel, cele două lumi fiind separate de apă și conectate de un pod. Utilizatorul are posibilitatea de a controla scena prin intermediul mouse-ului și a tastaturii.

2. Scenariul

2.1. Descrierea scenei și a obiectelor

Scena completă:

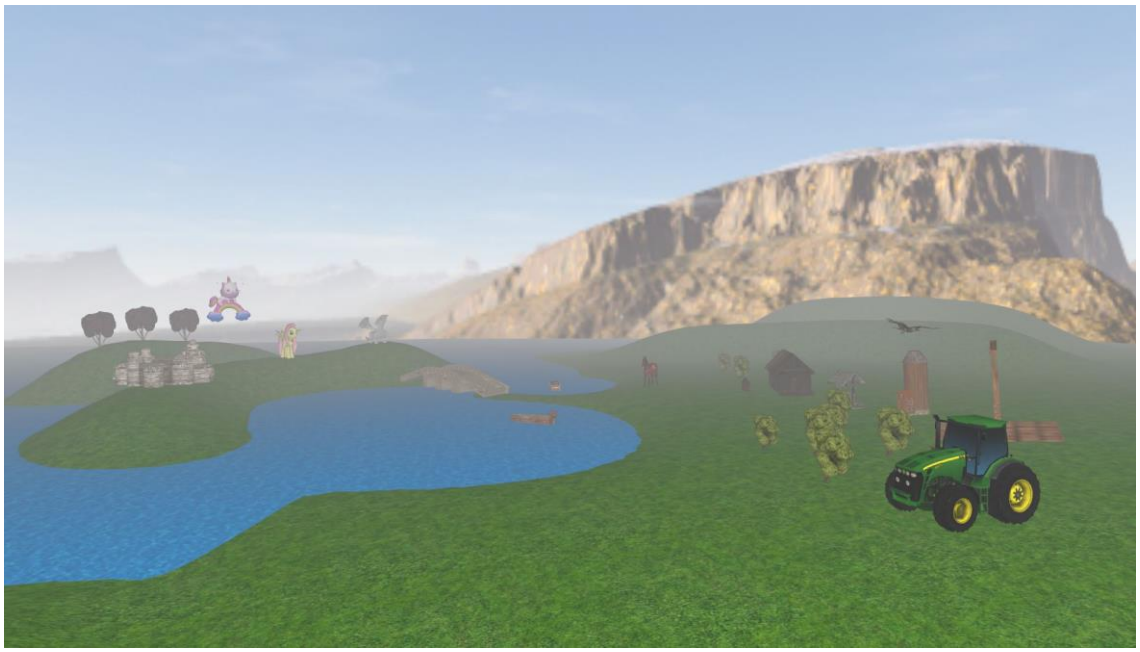
Într-o parte este lumea reală cu obiecte precum casă, fântână, hambar, moară de vânt, cușcă pentru câine, tractor, cal, pisică, câine, pasăre, copaci reali.

Cele două lumi sunt separate de apă, unde există bărci și sunt conectate de un pod.

În cealală parte este lumea fantastică cu obiecte precum castel, statuie, ponei, curcubeu imaginar, copaci mov.

2.2. Funcționalități

1. Scena poate fi vizualizată prin mișcare la stânga, dreapta, sus, jos, înainte, înapoi și prin rotație, utilizându-se tastatura, dar și mouse-ul. De asemenea, există o animație de prezentare, la apăsarea tastei G, scena rotindu-se pentru o vizualizare complete.
2. Se poate naviga între modurile de vizualizare solid (implicit), wireframe, poligonal, utilizând tastele Z și X.
3. Pasărea din peisajul real zboară la apăsarea tastei P.
4. Curcubeul imaginar se mișcă spre stânga sau spre dreapta la apăsarea tastelor L și K.
5. Se poate activa și dezactiva ceața folosind tasta F.



6. Se poate trece în modul noapte sau reveni în modul zi prin apăsarea tastei N.

3. Detalii de implementare

3.1. Funcții și algoritmi

3.1.1. Soluții posibile

Acest program OpenGL creează o fereastră, încarcă și afișează modele 3D într-un mediu tridimensional. Programul folosește biblioteca GLFW pentru gestionarea ferestrei și evenimentelor de la tastatură și mouse, iar biblioteca GLEW este utilizată pentru încărcarea funcțiilor OpenGL. De asemenea, programul utilizează biblioteca GLM pentru operații matematice în spațiul tridimensional.

Cu ajutorul clasei Camera este posibilă deplasarea prin scenă, înainte, înapoi, la stânga, la dreapta, în sus sau în jos, în funcție de tastele apăsate, precum și rotirea scenei. Obiectele au fost animate folosind operații precum translație și rotație și modificând matricele lor model. Aceste funcționalități permit utilizatorului să exploreze și să interacționeze cu scena 3D, oferind un control versatil asupra camerei și a obiectelor animate.

1. Clasa Camera este responsabilă pentru gestionarea matricei de vizualizare și actualizarea parametrilor internei ai camerei în funcție de intrarea utilizatorului prin mișcare și rotație. Principalele funcții:
    - Camera::Camera(glm::vec3 cameraPosition, glm::vec3 cameraTarget, glm::vec3 cameraUp): Constructorul clasei Camera, care inițializează poziția camerei, ținta și direcția sus și calculează direcțiile frontale și laterale.
    - glm::mat4 Camera::getViewMatrix(): Returnează matricea de vizualizare folosind funcția glm::lookAt(), care construiește o matrice de vizualizare pentru o cameră poziționată la cameraPosition,, care privește către cameraTarget, cu direcția specificată de cameraUpDirection.
    - void Camera::move(MOVE_DIRECTION direction, float speed): Actualizează poziția și ținta camerei în funcție de direcție și viteză. Mutările includ înainte, înapoi, stânga, dreapta, sus și jos.
    - void Camera::rotate(float pitch, float yaw): Actualizează parametrii camerei după un eveniment de rotire a camerei, cum ar fi mișcarea mouse-ului. Se realizează o conversie a unghiurilor în radiani și se calculează noile direcții ale camerei.

2. Main:
    - void windowResizeCallback(GLFWwindow* window, int width, int height): Callback pentru redimensionarea ferestrei, actualizează matricea de proiecție și setează viewport-ul.
    - void keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode): Callback pentru evenimentele tastaturii, gestionează diverse acțiuni, cum ar fi mișcarea camerei, activarea/dezactivarea unor efecte și schimbarea modului de afișare.
    - void mouseCallback(GLFWwindow* window, double xpos, double ypos): Callback pentru evenimentele mouse-ului, gestionează mișcarea mouse-ului și realizează rotația camerei în funcție de aceasta.
    - void updateShaderUniforms(): Actualizează variabilele uniforme ale shader-ului cu matricele de model și vedere și matricea normală.

- void processMovement(): Gestionează deplasarea camerei și actualizează matricea de vedere în funcție de tastele apăsate.
- void initOpenGLWindow(): Inițializează fereastra OpenGL.
- void setWindowCallbacks(): Setează callback-urile pentru ferestra GLFW.
- void initOpenGLState(): Inițializează starea OpenGL, inclusiv culoarea de fundal, testul de adâncime și modul de afișare a poligoanelor.
- void initModels(): Încarcă modelele 3D folosite în program.
- void initShaders(): Încarcă și compilează shader-ele folosite în program.
- void initUniforms(): Inițializează variabilele uniforme ale shader-ului, cum ar fi matricele de model, vedere și proiecție, direcția luminii, culoarea luminii.
- void renderScena(gps::Shader shader): Afișează scena principală.
- void renderObiect(gps::Shader shader): Afișează un obiect.
- void renderObiectDeMiscat(gps::Shader shader): Afișează un obiect care poate fi mișcat în jurul axei Y.
- void renderScene(): Afișează întreaga scenă, inclusiv obiecte 3D și skybox.
- void updateObiectDeMiscat(): Actualizează modelul obiectului care se poate mișca în funcție de unghiul de rotație.
- void rotate(): Realizează o rotație a scenei în jurul axei Y.
- void initSkyBox(): Inițializează skybox-ul.
- void cleanup(): Eliberează resursele utilizate de fereastră.

3. SkyBox:
    - void Load(std::vector<const GLchar*> faces): Încarcă texturile pentru cele 6 fețe ale skybox-ului.
    - void Draw(gps::Shader shader): Afișează skybox-ul folosind o matrice de vizualizare și o matrice de proiecție.

4. Alte funcții:
    - glCheckError(): Funcție auxiliară pentru a afișa erori OpenGL în consolă.
    - glfwPollEvents(): Funcție folosită pentru a procesa toate evenimentele din coada de evenimente GLFW.
    - glfwSwapBuffers(myWindow.getWindow()): Funcție care face schimbul de buffer-e pentru fereastra GLFW.

3.1.2. Motivarea abordării alese

Codul folosește biblioteci populare precum GLFW, GLEW și GLM, ceea ce face gestionarea ferestrei și a funcțiilor OpenGL mai ușoară și mai accesibilă. Structura codului, cu clase separate pentru cameră și skybox ajută la organizarea și modularizarea codului. Implementarea unei bucle principale și a funcțiilor de desenare oferă o structură clară pentru un program OpenGL. Programul acoperă mai multe funcționalități precum gestionarea evenimentelor de la tastatură, mouse, desenarea obiectelor 3D și a unui skybox, mișcarea unor obiecte, trecerea în mod noapte, apariția fenomenelor precum ceață.

3.2. Modelul grafic

Scena a fost modelată în Blender. Terenul (câmpul) și apa au fost proiectate de către mine în Blender, iar obiectele și texturile au fost descărcate de pe internat și apoi poziționate în modul dorit folosind această aplicație. Am exportat scena in formatul .obj și am încarcat-o în proiectul OpenGl. Texturile au fost aplicate folosindu-se fișierul .mtl. Am exportat separat doar unele obiecte, de exemplu pe cele pe care am dorit să le animez (curcubeul imaginar și pasărea). Pentru cer, am adăugat în proiect SkyBox-ul prezentat în laborator.

3.3. Structuri de date

În proiect au fost utilizate clase, precum Camera și Window, structuri de date GLM (glm::vec3, glm::mat3, glm::mat3), obiecte de tipul Model3D, obiect de tipul gps::Window care gestionează fereastra OpenGl, array de bool-uri pentru a ține evidența tastelor apăsate (pressedKeys[]), obiecte de tip shader (gps::Shader), obiect de tipul SkyBox (gps::SkyBox).

3.4. Ierarhia de clase

1. Main – aici se creează fereastra, se implementează logica pentru gestionarea evenimentelor și a erorilor, se creează scena, se realizează animațiile, folosind clasele precizate mai jos.
2. Camera – conține metode pentru manipularea mișcării și rotației camerei într-un mediu 3D, folosind biblioteca GLM, încapsulând metode comune precum deplasarea înainte, înapoi, la stânga, la dreapta, în sus și în jos și rotația spre stânga și spre dreapta.
3. Mesh – clasă pentru a reprezenta un obiect tridimensional într-un mediu OpenGl. Datele sunt stocate eficient în buffer-e pentru a fi transferate rapid către unitatea de procesare grafică. De asemenea, permite afișarea mesh-urilor cu texturi și iluminare folosind un shader specificat.
4. Model3D – clasă pentru încărcarea și desenarea de modele 3D într-un mediu OpenGl. Gestionează corect texturile și realizează operațiile necesare pentru transferul acestora în memorie.
5. Shader – clasă pentru încarcarea și utilizarea de shadere într-un program OpenGl, care gestionează detaliile încărcării și compilării pentru shadere.
6. SkyBox – clasă utilizată pentru afișarea unui skybox în programul OpenGl. Skybox-ul este utilizat pentru a crea un fundal aparent infinit într-o scenă 3D. Este desenat în jurul camerei pentru a crea impresia că obiectele se află într-un spațiu foarte mare.
7. Window – clasă utilizată pentru gestionarea ferestrei într-o aplicație OpenGl, utilizând GFLW. Este interfața pentru gestionarea ferestrei în aplicație și oferă funcționalități esențiale, precum crearea, distrugerea și obținerea dimensiunilor ferestrei.

4. Prezentarea interfeței grafice utilizator

Utilizatorul poate interacționa cu scena astfel:

W – pentru a mișca înainte
S – pentru a mișca înapoi
A - pentru a mișca la stânga
D - pentru a mișca la stânga

Q – pentru a roti spre stânga
E – pentru a roti spre dreapta
U – pentru a mișca în sus
Y – pentru a mișca în jos
G – pentru a porni animația scenei (se va roti)
F – pentru a porni/opri ceața
N – pentru a intra în modul noapte/reveni în modul zi
L – pentru a mișca curcubeul imaginar spre dreapta
K – pentru a roti curcubeul imaginar spre stânga
P – pentru a face pasărea să zboare
Z – pentru a vizualiza scena în modul wireframe
X – pentru a vizualiza scena în modul poligonal

5. Concluzii și dezvoltări ulterioare

În concluzie, proiectul a reprezentat o oportunitate bună pentru explorarea și aplicarea cunoștințelor de grafică pe calculator și programare OpenGL, precum și de a mă familiariza cu aplicația Blender, unde am modelat scena.

Posibile îmbunătățiri ar fi adăugarea mai multor surse de lumină și a umbrelor pentru a îmbunătăți aspectul vizual al scenei, extinderea interacțiunii utilizator prin implementarea de funcționalități precum selecția obiectelor, manipularea obiectelor și adăugarea mai multor animații.

6. Referințe

1. Tutoriale Blender, https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM
2. Curs Moodle - Prelucrare Grafică, Seria A, Sem. 1, https://moodle.cs.utcluj.ro/course/view.php?id=612
3. https://free3d.com/
4. https://sketchfab.com/search?q=rainbow&sort_by=-likeCount&type=models
5. https://learnopengl.com/Getting-started/OpenGL