UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Marian Iosif Andrei
Grupa: 30232

Village – prelucrare grafica

1. Cuprins

2. Prezentarea temei

Tema proiectului constă în dezvoltarea unui joc în mediul OpenGL, care reprezintă un sat virtual. Scena este compusă din diverse elemente, precum teren cu iarba, dealuri, un lac, case, o căruță, copaci, tufișuri, felinare, garduri și porți. Scopul jocului este să ofere o experiență interactivă utilizatorilor, permitându-le să exploreze și să interacționeze cu mediul virtual.

3. Scenariul

3.1. Descrierea Scenei și a Obiectelor

- **Terenul**: Un teren amplu cu texturi de iarbă, dealuri și un lac.
- **Căruța**: O căruță statică sau în mișcare, care poate fi observată și controlată de utilizator. Acesta poate face ca, căruța să se miște în față sau în spate folosind tastele E și Q. Totuși aceasta nu poate ieși din sat, nu poate depăși barierele de care e delimitat satul (nu poate trece de gard)
- **Case**: Clădiri reprezentând case tradiționale în sat, aceste obiecte sunt statice, fac parte din scenă și sunt puse pentru a exemplifica maparea texturilor si complexitatea acesora, separarea obiectelor care apartin casei, pe texturi.
- **Copaci și Tufișuri**: Reprezintă alte obiecte statice care sunt folosite pentru fotorealism, fapt pentru care am implementat și algoritmul de reducere a fragmentelor care reduce din textura originală a frunzei doar imaginea frunzei fară conturul acesteia.
- **Felinare**: Felinarele sunt alte obiecte care fac parte din décor, dar care ajută la relismul scenei, pentru a putea plasa luminile punctiforme in locuri unde sa nu apară doar așa de nicasieri si sa faca parte din ceva realistic.
- **Garduri și Porți**: Elemente de delimitare a satului de restul scenei
- **Ceainicul**: Acesta este folosit in scena pentru exemplificarea prezenței luminii globale in proiect, in acesta se poate observa strălucirea luminii globale si reflexia acesteia in obiect.
- **Zepelinul:** Acest obiect se misca dar nu poate fi controlat de utilizator, acesta se invarte in jurul centrului in mod constant, iar directia obiectului se roteste si ea in functie de miscarea acestuia.

3.2. Funcționalități

- **Explorare**: Utilizatorul poate explora satul într-un mod interactiv, schimbând perspectivele și orientarea camerei. Utilizatorul poate folosi atat tastele tastaturii cat si mouse-ul pentru a-si misca pozitia si perspectiva camerei asupra scenei. Butoanele “A”, “S”, “D”, “W” detin controlul caracterului din joc, comenzi pentru a

merge in stanga, spate, dreapta, respectiv inainte. Simpultan acesta poate controla privirea asupra scenei cu ajutorul mouse-ului.

- **Interacțiune cu Obiectele**: Posibilitatea de a interacționa cu obiectele, exista cateva taste pe care daca utilizatorul le apasa, acesta va genera actiuni asupra unor obiecte din scena, cum ar fi: pentru controlul carutei, acesta poate apasa butoanele “Q” si “E” care fac ca obectul sa se deplaseze pe axa X in spate si in fata. Pe tastele “B” si “N” utilizatorul interactioneaza cu ceainicul si il poate face sa se invarta in ambele sensuri si poate vedea inca reflexia luminii in acesta

```
model = glm::translate(glm::mat4(1.0f), glm::vec3(distX, 0.0f, 0.0f));
glUniformMatrix4fv(glGetUniformLocation(shader.shaderProgram, "model"), 1, GL_FALSE, glm::value_ptr(model));
if (!depthPass) {
    normalMatrix = glm::mat3(glm::inverseTranspose(view * model));
    glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));
}
cart.Draw(shader);


glm::vec3 centerPoint(40.397f, 5.23f, -45.0f);
model = glm::translate(glm::mat4(1.0f), glm::vec3(40.397f + distX, 5.23f, -45.0f));
model = glm::rotate(model, glm::radians(angleX), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, -centerPoint);
glUniformMatrix4fv(glGetUniformLocation(shader.shaderProgram, "model"), 1, GL_FALSE, glm::value_ptr(model));
if (!depthPass) {
    normalMatrix = glm::mat3(glm::inverseTranspose(view * model));
    glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));
}
cart_wheels.Draw(shader);
```

Figura 1: Exemplificare animatiei căruței

- **Controlarea vremii:** Utilizatorul poate controla cand sa se faca zi sau noapte apasand butonul “X”, asa vor fi exemplificate mai usor luminile punctiforme, iar prin apasarea tastei “Z” el poate genera o ploaie care se randeaza aleator.

- **Animare**: Elemente animate, precum roțile căruței acestea se invartesc realist pe drum in tim ce se deplaseaza inainte si inapoi odata cu caruta. Animatia zepelinului este una statica, acesta se invarte cu o viteza constanta in jurul originii.

- **Alte functionalitati:** Este posibilitate de a vedea scena si obiectele de pe aceasta sub forma de wireframe sau point sau smooth, aceste moduri sunt diferite fata de modul poligonal setat ca default, aceste functionalizati se pot genera folosind tastele “T”, “Y”, “U”. O alta funtionalitate este cea de prezentare, in care utilizatorul apasa tasta “L” pentru care se face o scurta prezentare a scenei si a functionalitatilor acesteia.

4. Detalii de implementare:

4.1. Funcții și Algoritmi:

4.1.1. Soluții Posibile

Explorare și Mișcare:

Pentru a permite utilizatorului să exploreze și să se miște liber prin sat, vom utiliza tastatura și mouse-ul. Implementarea va implica gestionarea input-ului tastelor

pentru a modifica poziția și orientarea camerei în funcție de comenzi precum 'W', 'A', 'S', 'D' pentru mișcare înainte, la stânga, înapoi și la dreapta.

Ciclu Zi-Noapte și Vreme Dinamică:

Implementarea ciclului zi-noapte și a vremii dinamice va implica utilizarea shaderelor și ajustarea parametrilor de iluminare. Shader-ele vor manipula culorile și intensitățile luminilor pentru a simula tranzitia realistă de la zi la noapte. De asemenea, vor exista variabile care controlează starea vremii și pot activa efecte precum ploaia.

```
if (itsRaining) {
    GLfloat x = rand() % 200 - 100;
    GLfloat z = rand() % 200 - 100;
    GLfloat x1 = rand() % 200 - 100;
    GLfloat z1 = rand() % 200 - 100;
    GLfloat x2 = rand() % 200 - 100;
    GLfloat z2 = rand() % 200 - 100;
    GLfloat x3 = rand() % 200 - 100;
    GLfloat z3 = rand() % 200 - 100;
    GLfloat x4 = rand() % 200 - 100;
    GLfloat z4 = rand() % 200 - 100;
    GLfloat x5 = rand() % 200 - 100;
    GLfloat z5 = rand() % 200 - 100;

    for (GLfloat y = 50.0f; y > -1.0f; y -= 15.0f) {
        renderRain(myCustomShader, x, y, z);
        renderRain(myCustomShader, x1, y, z1);
        renderRain(myCustomShader, x, y, x1);
        renderRain(myCustomShader, z, y, z1);
        renderRain(myCustomShader, x2, y, z2);
        renderRain(myCustomShader, x1, y, z2);
        renderRain(myCustomShader, x, y, x2);
        renderRain(myCustomShader, z, y, z3);
        renderRain(myCustomShader, x3, y, z1);
        renderRain(myCustomShader, x4, y, z2);
        renderRain(myCustomShader, x2, y, x3);
        renderRain(myCustomShader, z3, y, z3);
        renderRain(myCustomShader, x, y, z1);
        renderRain(myCustomShader, x2, y, z3);
        renderRain(myCustomShader, x3, y, x2);
        renderRain(myCustomShader, z3, y, z1);
        renderRain(myCustomShader, x4, y, z5);
        renderRain(myCustomShader, x5, y, z4);
        renderRain(myCustomShader, x3, y, x5);
        renderRain(myCustomShader, z5, y, z4);
        for (int j = 0; j < 100; j++) {
            j++;
            if (!itsRaining) break;
        }
```

Figura 2: Prezentarea algoritmului de generare a ploii random

Animare:

Pentru a anima obiecte precum roțile căruței, vom utiliza un schelet de animație. Acesta constă într-un set de oase și articulații care pot fi animate, permițând obiectelor să se miște fluid și natural.

4.1.2. Motivarea Abordării Alese:

Alegerea shaderelor și a funcțiilor avansate OpenGL este motivată de capacitatea lor de a furniza efecte realiste de iluminare și atmosferă. Shader-ele oferă control detaliat asupra aspectului vizual al scenei, permițându-ne să obținem o experiență vizuală captivantă pentru utilizator.

4.2. Modelul Grafic

Modelul grafic se bazează pe obiecte 3D create în Blender, asigurându-ne că acestea sunt compatibile și pot fi încărcate eficient în scena OpenGL. Shader-ele sunt esențiale pentru adăugarea de iluminare și efecte speciale, precum schimbarea dinamică a culorilor în funcție de ciclul zi-noapte sau vremea simulată.

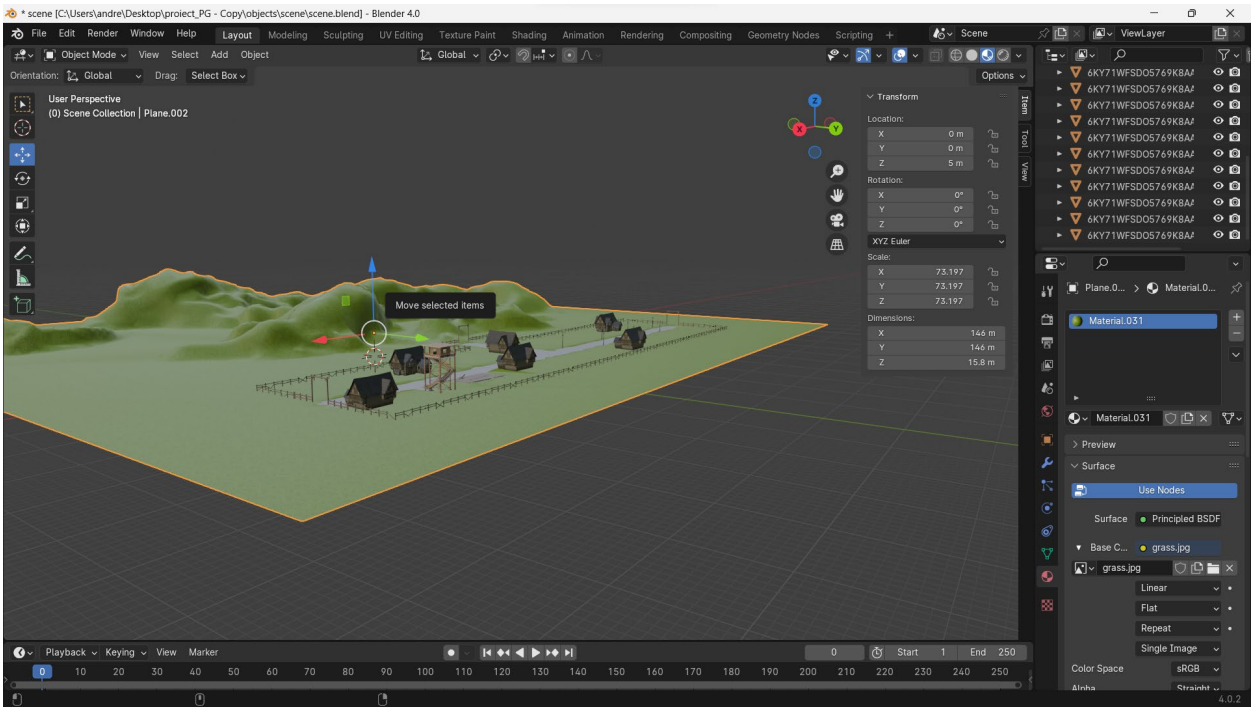

Figura 3: Prezentarea scenei modelata in Blender

4.3. Structuri de Date

Utilizarea structurilor de date este necesară pentru stocarea informațiilor despre obiecte și gestionarea interacțiunilor. Fiecare obiect din scenă va avea atribute precum poziție, dimensiune și texturi, stocate eficient pentru a permite o manipulare ușoară.

4.4. Ierarhia de Clase

Ierarhia de clase este esențială pentru organizarea eficientă și modulară a funcționalităților. Vom avea clase distincte pentru camere, obiecte 3D, lumini și particule. Aceasta permite o gestionare mai ușoară a fiecărui tip de obiect și oferă extensibilitate pentru adăugarea ulterioară a altor tipuri de obiecte în scenă.

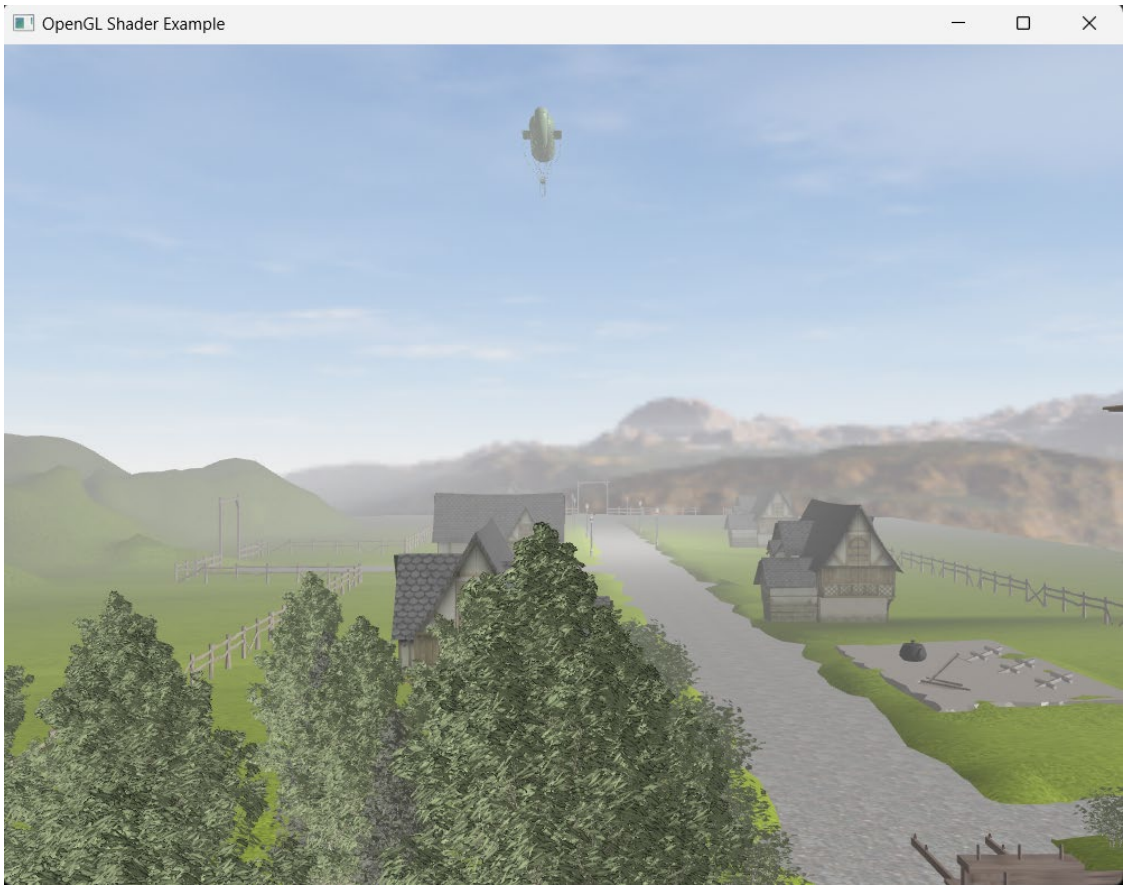

Figura 4: Scena de obiecte si obiectele randate in program

5. Prezentarea Interfeței Grafice Utilizator / Manual de Utilizare:

5.1. Descrierea Funcției processMovement()

Funcția processMovement() este esențială pentru interacțiunea utilizatorului cu jocul. Aceasta este apelată în mod regulat pentru a gestiona mișcarea și acțiunile în funcție de tastele apăsate de către utilizator. Vom detalia comportamentul acestei funcții în ceea ce privește fiecare tastă în parte.

5.2. Tastatura și Mișcarea

W - Mișcare înainte

Când utilizatorul apasă tasta 'W', funcția processMovement() ajustează poziția camerei în direcția înainte. Aceasta poate implica modificarea coordonatelor x, y și z ale camerei pentru a simula deplasarea înainte.

A - Mișcare la Stânga

Apăsarea tastei 'A' determină mișcarea camerei la stânga. Coordonatele camerei sunt ajustate astfel încât utilizatorul să se deplaseze lateral la stânga în cadrul scenei.

S - Mișcare Înapoi

Tasta 'S' are ca rezultat mișcarea camerei înapoi. Similar cu mișcarea înainte, coordonatele camerei sunt modificate pentru a simula deplasarea înapoi în spațiu.

D - Mișcare la Dreapta

Mișcarea la dreapta este activată prin apăsarea tastei 'D'. Aceasta ajustează coordonatele camerei pentru a permite deplasarea laterală la dreapta.

5.3. Interacțiune cu Obiectele

Funcția processMovement() poate, de asemenea, include logica pentru interacțiunea cu obiectele din joc. De exemplu, atunci când utilizatorul se află în apropierea unui obiect și apasă o tastă specifică (de exemplu, tasta 'E'), pot fi declanșate evenimente precum deschiderea unei uși sau colectarea unui obiect.

5.4. Alte Tastaturi și Acțiuni

Alte taste pot fi asignate pentru diferite acțiuni, cum ar fi sărirea, aplecarea sau activarea unor abilități speciale. Fiecare acțiune asociată tastelor va fi gestionată în cadrul funcției processMovement(), asigurându-se că interacțiunea utilizatorului cu jocul este fluidă și intuitivă.

6. Concluzii și Dezvoltări Ulterioare:

Proiectul încheiat reprezintă o realizare semnificativă, furnizând o experiență captivantă și interactivă pentru utilizatori. Integrarea shaderelor și funcțiilor OpenGL avansate a contribuit la obținerea unor efecte realiste de iluminare și atmosferă, consolidând astfel calitatea vizuală a jocului. Dezvoltările ulterioare pot viza diversificarea experienței prin adăugarea de noi obiecte și caracteristici, crescând astfel complexitatea și implicarea utilizatorilor. Optimizarea performanței se află, de asemenea, pe lista de priorități, pentru a asigura o rulare fluentă și eficientă a aplicației. Extinderea funcționalităților interactive poate include implementarea unor mecanici de joc inovatoare, permitând utilizatorilor să exploreze și să interacționeze într-un mod mai variat și captivant. În concluzie, proiectul actual oferă o bază solidă pentru viitoare îmbunătățiri și inovații, deschizând calea către o experiență de joc din ce în ce mai atractivă și complexă.

7. Referințe:

- OpenGL Documentation: https://www.opengl.org/documentation/
- GLFW Documentation: https://www.glfw.org/documentation.html
- Blender Documentation: https://docs.blender.org/manual/en/latest/