UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Prelucrare Grafica

Documentatie Proiect

Autor: Popovici Eusebiu-Ionut

Grupa: 30234

Facultatea de Automatica si Calculatoare

15 Ianuarie 2024

Cuprins

1 Prezentarea temei **2**

2 Scenariul **2**
2.1 Descrierea scenei si a obiectelor 2
2.2 Functionalitati 5

3 Detalii de implementare **6**
3.1 Functii,algoritmi,solutii si motivare: 6
3.2 Modelul grafic 9
3.3 Structuri de date 9
3.4 Ierarhia de clase 9

4 Prezentarea interfeței grafice utilizator **10**

5 Concluzii și dezvoltări ulterioare **10**

6 Referințe **10**

1 Prezentarea temei

Pentru acest proiect am decis sa creez o ferma intr-o zona muntoasa.Timpul trece în scenă astfel încât putem să o vedem pe parcursul zilei și al nopții. Cu fundalul și iarba de pe sol, am vrut să o fac să arate ca o zi de vară.Pe aceasta zona muntoasa putem observa o multime de copaci,niste animale,un lac si un munte,iar in acest mod am alcatuit ferma mea.

2 Scenariul

Scena arata ca o ferma intr-o zona de munte.Aici avem un cal care se poate misca,o casa aflata intre copaci,o barca si un mini tarc cu vaci.Peste lac putem observa un pod care duce la o moara de vant.

2.1 Descrierea scenei si a obiectelor

.

Figura 1: Scena

.

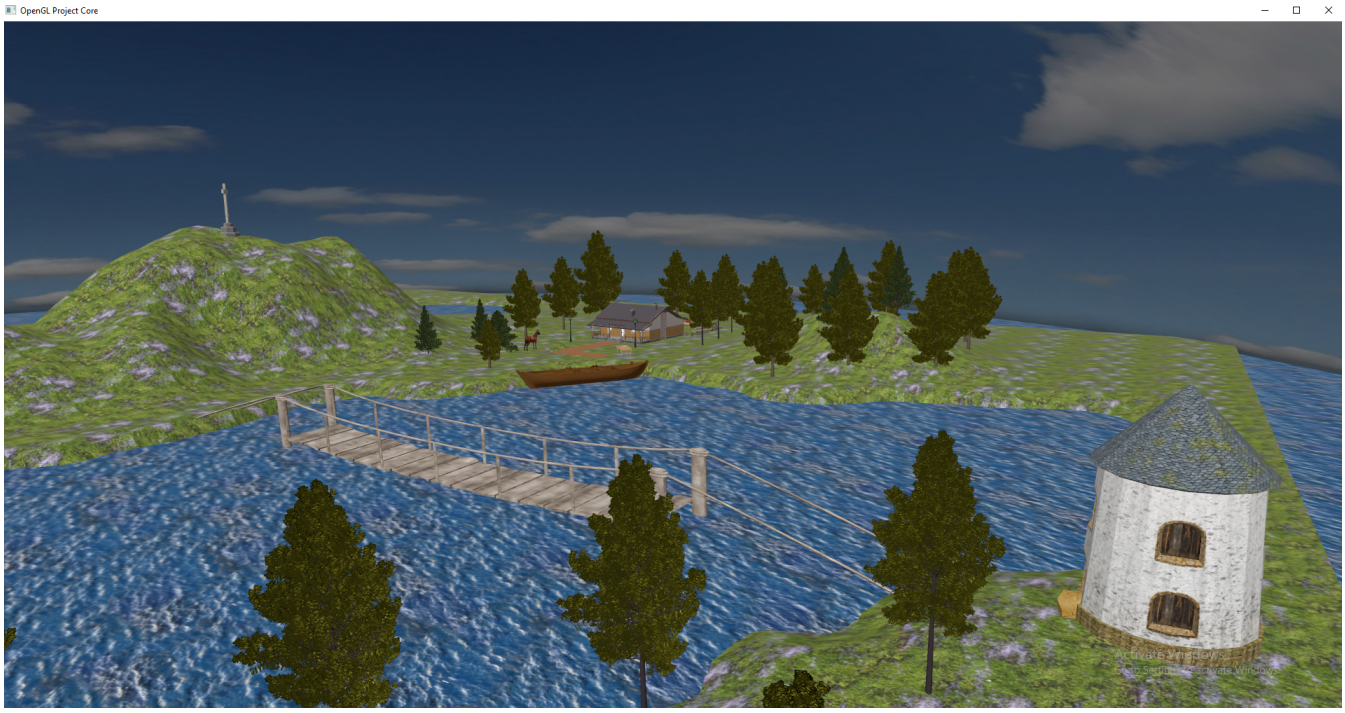

Figura 2: Full Scena

În prim-plan, observăm o canoe maro pe un râu cu apă albastră, care reflectă lumina, sugestiv pentru mișcarea apei. Deasupra apei, pe mal, se află o pășune verde cu pietre, pentru a sugera o zonă sălbatică,muntoasa.

În fundal, putem vedea o fermă, cu un conac clasic de culoare bej, acoperit cu țigle maro, și o verandă în față. Lângă casă, sunt câteva elemente de mobilier de grădină și o lampă de stradă, sugerând o încercare de a crea un mediu rural detaliat. Mai departe de casă, un cal maro se află lângă câțiva copaci înalți, care par să fie foșnetiți de vânt, încadrând scena și adăugând o atmosferă calmă, pastorală. Cerul este parțial înnorat, sugerând posibil o schimbare a timpului, de la o zi însorită la o seară înnorată.

- Scena de bază este alcătuită din două planuri texturate: unul care reprezintă apa și altul care reprezintă iarba. Planul apei a fost texturat în așa fel încât să capteze proprietățile optice ale apei, cum ar fi reflexia și refracția luminii, pentru a simula mișcarea și strălucirea apei reale.Planul ierbii a fost, de asemenea, texturat pentru a reflecta aspectul unui teren acoperit cu iarbă. Textura ierbii ar putea include variații de culoare pentru a simula diferite lungimi și tipuri de vegetație.
- Textura cerului, cunoscută în terminologia graficii 3D drept SkyDome, a fost creată folosind software-ul de modelare 3D Blender. SkyDome-ul este realizat prin prelucrarea părții superioare a unei sfere, proces în care normalele acesteia (adică direcțiile perpendiculare pe suprafața fiecărui poligon) au fost inversate pentru a se orienta spre interiorul sferei în loc de exterior, acest lucru fiind o practică obișnuită pentru a crea iluzia unui cer înconjurător într-un mediu 3D. Peste această sferă modificată, a fost aplicată o textură specifică, care să reprezinte cerul, posibil cu nori sau alte elemente vizuale care contribuie la atmosfera generală a scenei.

.

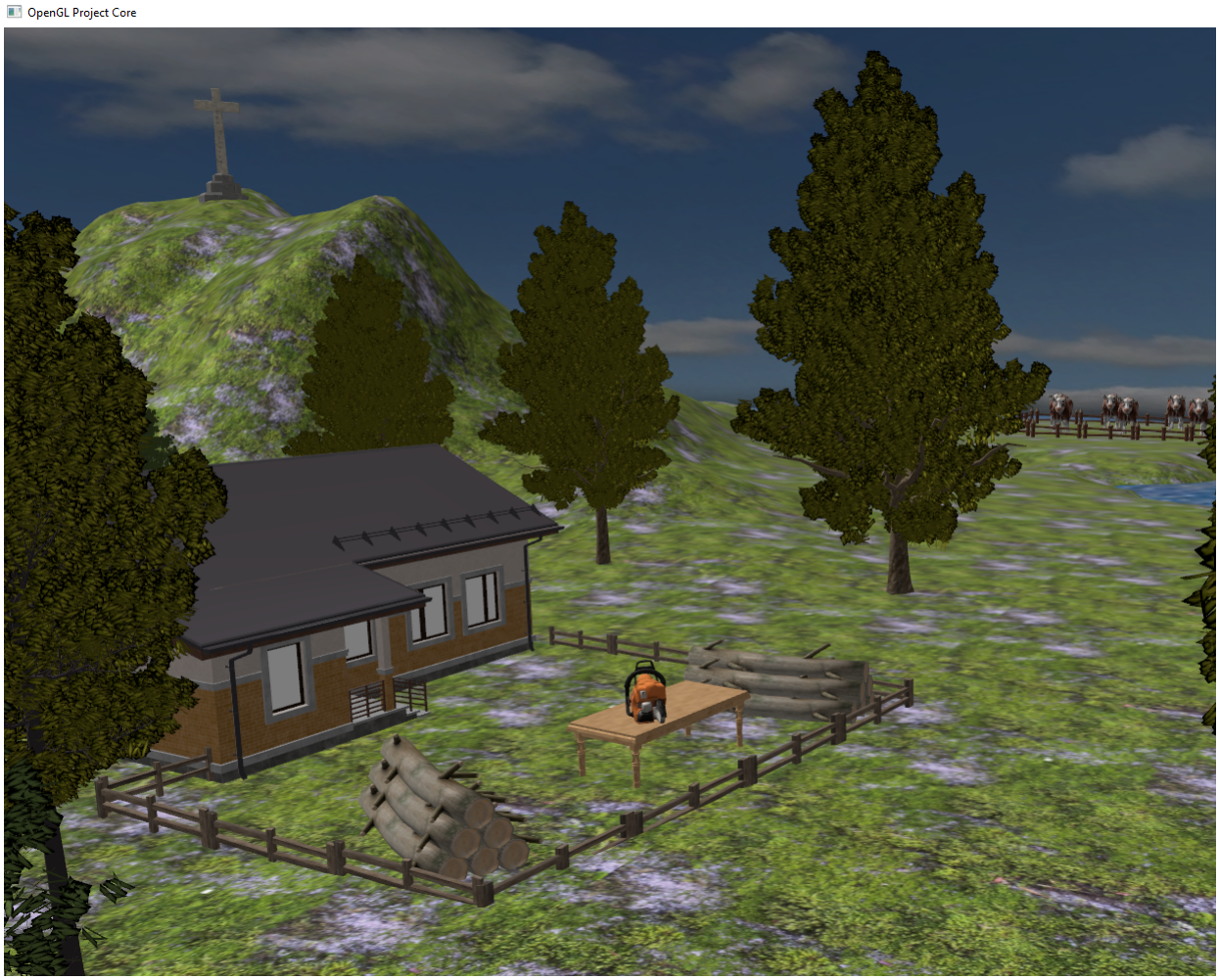

Figura 3: Scena lateral

- În jurul conacului, care este un singur obiect în contextul modelării 3D, observăm detalii care aduc mai multă autenticitate și complexitate scenei. Unul dintre aceste elemente este un garduleț. Acesta a fost creat prin multiplicarea unui singur obiect de model 3D
- Pe lângă gard, scena include și niște lemne proaspat tăiate, care sunt dispuse într-o manieră care să pară că au fost tocmai pregătite pentru foc sau pentru alte treburi gospodărești. Acestea pot avea diferite dimensiuni și forme pentru a sugera că tăierea a fost făcută manual.Și drujba, lăsată pe o masă, contribuie la narativa scenei, sugerând activitate recentă. În termeni de modelare 3D, drujba este probabil un model detaliat care include aspecte precum lama, corpul și chiar mânerul, toate texturate pentru a reflecta materialitatea

.

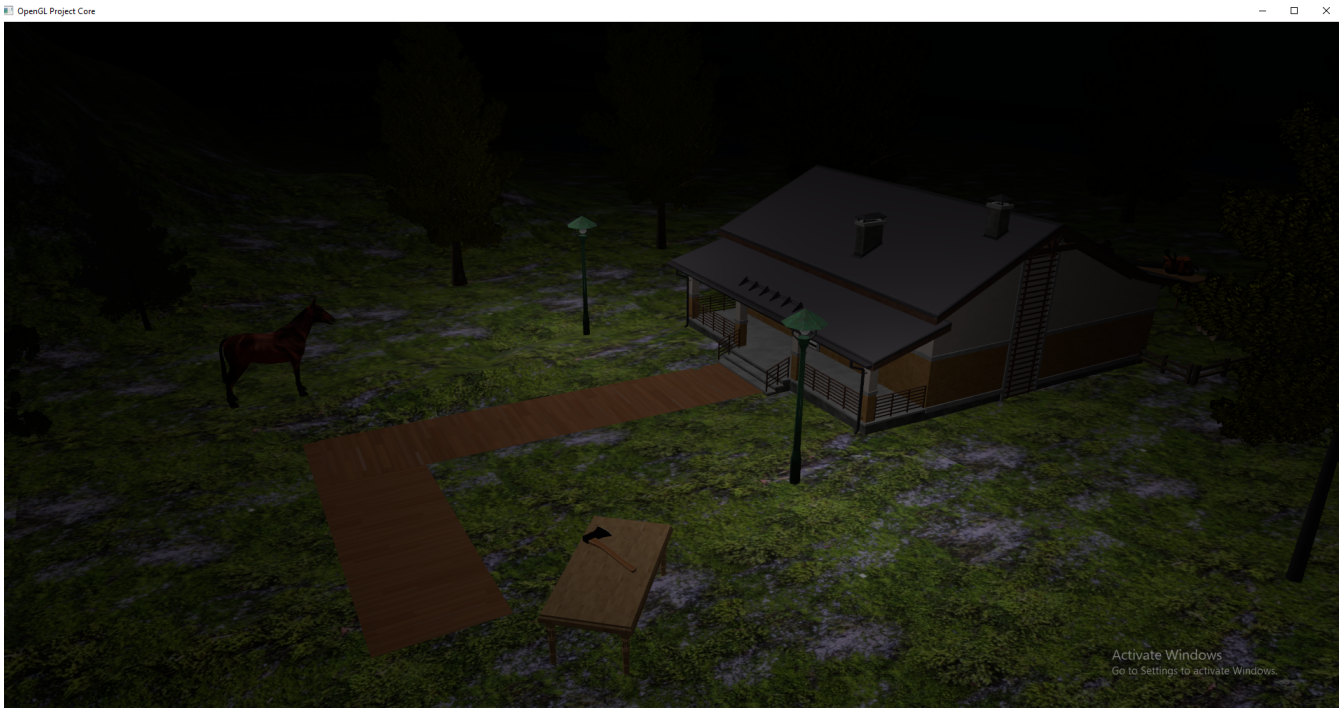

Figura 4: Lumini punctiforme

- Această imagine ilustrează o scenă nocturnă a unei ferme, creând o atmosferă diferită față de luminozitatea diurnă. Focalizarea este pe o casă cu structură simplă, iluminată parțial de lumina slabă a serii. În jurul casei, întunericul este străpuns de două felinare, fiecare emanând o lumină punctiformă - o formă de iluminare care emană în toate direcțiile de la un singur punct. Aceste lumini punctiforme creează halouri de lumină în jurul lor, contrastând cu întunericul dens din restul scenei și evidențiind porțiunea de teren pe care sunt plasate,activandu-se apasand tasta "K".Această iluminare artificială este în contrast marcant cu lumina direcțională a zilei, care de obicei bănește scena cu o lumină uniformă și difuză, simulând soarele. Totodata,pe o masă din fața casei, o toporișcă este lăsată, poate indicând întreruperea unei activități zilnice la venirea serii.

2.2 Functionalitati

- Vizualizarea scenei: toate obiectele din scenă sunt aranjate și plasate folosind translația, scalarea și rotația.
- Mișcarea camerei: camera se mișcă în scenă în 4 direcții folosind intrarea de la tastatură și poate, de asemenea, să se rotească pe 2 axe utilizând tastatura. Mișcarea este restricționată la o zonă specificată, astfel încât camera nu poate ieși din scenă pe axele x și z. Dar poate să se miște în sus și în jos. Camera se mișcă și prin animație. Avansează până când atinge granița. În acel moment se rotește până când poate să se miște liber.
- Specificarea surselor de lumină: au fost specificate 2 surse de lumină, lumină direcțională (soare) și lumină de tip spot(cele 2 felinare).
- Vizualizarea suprafețelor: Obiectele pot fi văzute ca lucruri solide (nu se poate vedea prin ele) sau ca structură de sârmă (doar marginile sunt vizibile). În scenă există suprafețe poligonale (solul, frunzele copacilor, casa, moara de vânt) și suprafețe netede (calul,vacile,felinarele).
- Textura si maparea materialelor: Toate obiectele au mapate texturi,unele din ele fiind mapate si redimensionate de catra mine,iar celelalte donwloadate ok.

- Animatia unor obiecte: Calul se poate misca fata,spate,stanga si dreapta,dar chiar si sus si jos pentru a simula urcarea pe un deal.Totodata,elicea de la moara de vant se poate roti in jurul originii sale,doar ca am intampinat probleme la ducerea ei in originea originala.

3 Detalii de implementare

Acest capitol va descrie algoritmii si functiile pe care le-am folosit pentru a detalia proiectul in care se afla scena mea.

3.1 Functii,algoritmi,solutii si motivare:

- **Trecerea timpului si aparitia noptii:** pentru aceasta metoda am folosit urmatoarea solutie: .

```
if (pressedKeys[GLFW_KEY_N]) {
    if (red >= 0.0f && green >= 0.0f && blue >= 0.0f) {
        red -= 0.005f;
        green -= 0.005f;
        blue -= 0.005f;
    }
}

if (pressedKeys[GLFW_KEY_B]) {
    if (red <= 1.0f && green <= 1.0f && blue <= 1.0f) {
        red += 0.005f;
        green += 0.005f;
        blue += 0.005f;
    }
}
```

Figura 5: Cod zi-¿lumina

Aceste două blocuri de cod sunt condiții (if statements) care ajustează intensitatea culorilor roșu, verde și albastru (RGB).Primul if este activat atunci când utilizatorul apasă tasta 'N'. Condiția verifică dacă valorile pentru roșu, verde și albastru sunt toate mai mari sau egale cu 0.0f.Dacă este adevărat, acesta scade fiecare valoare de culoare cu 0.005f, ceea ce face ca scena să devină treptat mai întunecată.Al doilea if este activat când utilizatorul apasă tasta 'B'. Verifică dacă valorile pentru roșu, verde și albastru sunt toate mai mici sau egale cu 1.0f. Dacă condiția este îndeplinită,scena devine treptat mai luminată.

- **Folosirea luminilor din felinare:** pentru aceasta metoda am folosit urmatoarea solutie, implementad in 2 lumini punctiforme in fragment shader: .

```
position = glGetUniformLocation(myBasicShader.shaderProgram, "pos");
glm::vec3 pos1 = glm::vec3(9.584325f, 8.536662f, -95.028366f);
glUniform3fv(position, 1, glm::value_ptr(pos1));

position1 = glGetUniformLocation(myBasicShader.shaderProgram, "pos1");
glm::vec3 pos2 = glm::vec3(40.584325f, 8.536662f, -95.028366f);
glUniform3fv(position1, 1, glm::value_ptr(pos2));
```

Figura 6: Lumini punctiforme main

.

```
vec3 calcAttenuation(vec3 pos)
{
    float distance    = length(pos - fPosition);
    float attenuation = 1.0 / (ct + linear * distance +
                    quadratic * (distance * distance));
    ambient  *= attenuation;
    diffuse  *= attenuation;
    specular *= attenuation;

    return(ambient + diffuse + specular);
}
vec3 computePointLight(){
    vec4 fPosEye = view * model * vec4(fPosition, 1.0f);
    vec3 normalEye = normalize(normalMatrix * fNormal);

    vec3 lightDirN = vec3(normalize(view * vec4(lightDir, 0.0f)));

    vec3 viewDir = normalize(- fPosEye.xyz);

    ambient = ambientStrength * lightColor;

    diffuse = max(dot(normalEye, lightDirN), 0.0f) * lightColor;

    vec3 reflectDir = reflect(-lightDirN, normalEye);
    float specCoeff = pow(max(dot(viewDir, reflectDir), 0.0f), 32);
    specular = specularStrength * specCoeff * lightColor;

    return calcAttenuation(pos);

}
```

Figura 7: Lumini punctiforme fragment shader

Funcția calcAttenuation: Aceasta calculează atenuarea (slăbirea) luminii pe măsură ce se îndepărtează de sursa de lumină. Atenuarea depinde de distanța de la sursă la punctul de pe suprafață (distance) și de coeficienții constanți (ct), liniari (linear) și pătratici (quadratic) care definesc cum scade intensitatea cu distanța. Calculul atenuării se aplică la fiecare componentă de lumină - ambientală, difuză și speculară.Functia calcPointLight() este in mare ca cea de la directionala,doar ca ceea ce face in plus e calcularea componentei speculare,care se calculează prin reflectarea direcției luminii față de normala suprafeței și apoi compararea acestei direcții reflectate cu direcția de vizualizare folosind produsul scalar. Rezultatul este ridicat la puterea unui coeficient(32), care determină mărimea și intensitatea petei de lumină speculară. Culoarea speculară este produsul dintre această valoare speculativă, forța speculară și culoarea luminii.

- **Miscarea calului si rotatia elicei de la moara:** pentru aceasta metoda am folosit urmatoarea solutie, implementad 2 functii noi de moveHorse() si rotateWildmill(), functii care vor fi folosite in renderHorse() si renderWindimill(),importate ca obiecte diferite fata de scena pe care o avem deja. .

```
void moveHorse() { ... }
void renderHorse(gps::Shader shader) {
    // select active shader program
    shader.useShaderProgram();

    glm::mat4 horseTr = glm::translate(model, glm::vec3(horseX, horseY, horseZ));
    glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(horseTr));

    glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));

    horse.Draw(shader);
}
```

Figura 8: Animatie cal

.

```
float windmillRotationAngle = 0.0f;

void rotateWindmill() {
    if (pressedKeys[GLFW_KEY_R]) {
        windmillRotationAngle += 2.0f;
    }
}

void renderWindmill(gps::Shader shader) {
    // select active shader program
    shader.useShaderProgram();

    windmillRot = translate(glm::mat4(1.0f), glm::vec3(190.519f, 16.7732f, 47.3917f));
    windmillRot = glm::rotate(model, glm::radians(windmillRotationAngle), glm::vec3(1.0f, 0.0f, 0.0f));
    windmillRot = translate(windmillRot, glm::vec3(-190.519f, -16.7732f, -47.3917f));
    glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(windmillRot));

    glm::mat3 normalMatrix2 = (glm::inverseTranspose(view * windmillRot));
    glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix2));

    windmill.Draw(shader);
}
```

Figura 9: Animatie moara

Funcția renderHorse() activează un program shader specificat pentru a desena calul,calculează o matrice de transformare pentru cal, translându-l la coordonatele (horseX, horseY, horseZ),setează uniformele pentru matricea de model și matricea normală în shader,desenează calul folosind metoda Draw a obiectului.Variabila windmillRotationAngle este folosită pentru a stoca unghiul de rotație actual al morii de vânt.Funcția rotateWindmill() verifică dacă tasta 'R' este apăsată și, dacă este, mărește unghiul de rotație al morii de vânt, făcând-o să se rotească.Funcția renderWindmill() activează programul shader pentru moară de vânt,calculează matricea de transformare pentru moară de vânt, rotind-o în jurul unei axe și translând-o la o anumită poziție,calculează matricea normală pentru a asigura că luminile și umbrele sunt corecte după rotație,setează uniformele pentru matricea de model și matricea normală în shader si desenează moara de vânt cu noua sa rotație.Practic, acest cod este folosit pentru animația și redarea (rendering) a două obiecte 3D într-o scenă OpenGL, permițându-le să fie manipulate și vizualizate în timp real.

3.2 Modelul grafic

Modelul grafic folosit în implementare este cel definit de OpenGL.

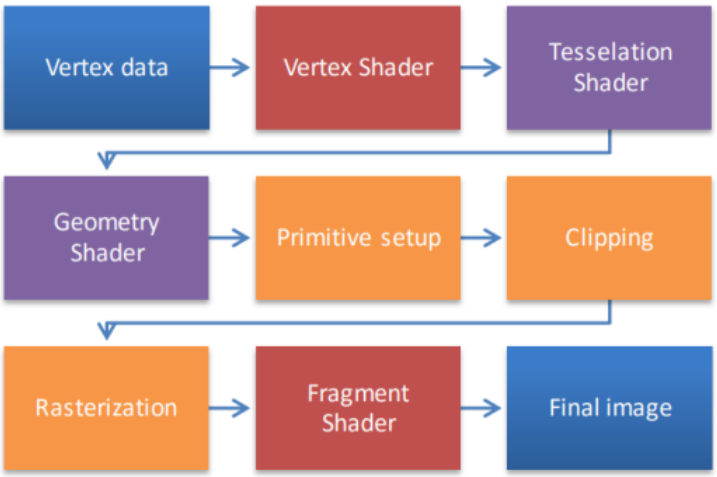

Figura 10: Model grafic

Nu am programat Shaderele de Teselare și Geometrie, doar Fragment Shader-ul si Vertex Shader-ul. În Shaderul de Fragment am efectuat majoritatea calculelor: lumină direcțională și de tip spot, ceață, incercare la umbre, texturi și culori. În Shaderul de Vârf am folosit matricea de transformare în spațiul luminii pentru a calcula poziția actuală a vârfului în spațiul luminii (scena văzută din perspectiva surselor de lumină) astfel încât să pot obține valoarea sa z, deoarece valorile din harta de adâncime erau în coordonatele luminii și aveam nevoie să fac o comparație pentru a calcula valoarea umbrei.

3.3 Structuri de date

Singurele structuri de date pe care le-am folosit au fost cele furnizate în șablonul de proiect. Singura pe care am modificat-o este clasa Cameră, în care am implementat metodele de mișcare și rotație și am adăugat câteva metode de accesare pentru poziția camerei, ținta și direcția din față, de care aveam nevoie pentru a face o copie a camerei pentru verificarea coliziunilor. De asemenea, aveam nevoie de direcția din față a camerei pentru calculul lanternei. Am adăugat și o funcție de previzualizare pentru a vedea întreaga scenă.

3.4 Ierarhia de clase

Ierarhia de clase a acestui proiect este destul de simpla si usoara de inteles:

- La nivelul cel mai înalt, avem main, care sugerează punctul de intrare principal al aplicației sau al jocului.
- Window ar putea fi componenta care gestionează crearea și manipularea ferestrei de vizualizare sau a contextului de redare.
- Model3D este cel mai probabil o clasă sau o structură care gestionează modelele tridimensionale care trebuie afișate.
- Camera gestionează poziționarea și orientarea punctului de vedere în scena 3D.

- Model3D este conectat la Shader, indicând că aceasta componenta foloseste shadere pentru a controla modul în care sunt desenate pe ecran. Shaderele sunt programe care rulează pe GPU pentru a realiza efecte vizuale în timp real.
- De asemenea, Shader este conectat direct la Camera, ceea ce sugerează că informațiile de vizualizare și proiecție furnizate de camera sunt utilizate în shadere pentru a determina cum sunt vizualizate obiectele din spațiul 3D.

4 Prezentarea interfeței grafice utilizator

Utilizatorul poate interactiona cu scena folosind tastatura si mouse-ul:

- Folosind "W","A","S" si "D" putem muta camera
- Folosind "Q" si "E" putem roti camera
- Putem folosi si mouse-ul pentru a ne schimba pozitia de vizulaizare in scena
- Folosind "F" putem activa ceata,iar cu G crestem densitatea acesteia
- Din sagetile "Front", "Back", "Right", "Left" putem misca calul sus, jos, dreapta si stanga,iar cu "5" si "2" il miscam sus si jos
- Din tastele "U", "I" si "O" putem facem trecerea de la modul wireframe, pointframe si solid mode
- De pe tastele "B" si "N" putem face zi/noapte
- De pe tasta "K" si "N" putem face trecerea intre cele 2 lumini
- De pe "ESC" putem opri aplicatia

5 Concluzii și dezvoltări ulterioare

In concluzie, acest proiect a fost foarte interesant si sunt bucuros ca am lucrat la el,invatand lucruri noi pe parcursul semestrului.Ca posibilitati de dezvoltare ulterioara s-ar putea enumera:

- Finalizarea umbrelor
- Mutarea elicei morii in originea ei corecta,pentru a fi rotita de acolo
- Scena ar putea fi mai complexa
- As putea implementa algoritmii de coliziune si detectie,pentru a creste realismul
- As putea baga modelul de iluminare Phong pe unele obiecte
- As mai putea adauga si efecte de ploaie,vant,ninsoare

6 Referințe

1. LearnOpenGL: Depth Testing
2. Free3D
3. RigModels
4. Stack Overflow: Rotating an Object Around a Fixed Point in OpenGL
5. LearnOpenGL: Multiple Lights
6. LearnOpenGL: Lighting Maps