Universitatea Tehnică din Cluj-Napoca
Departamentul de Calculatoare

Prelucrare Grafică

Proiect de semestru

Student: Lazea Dragoș-Bogdan
Grupa: 30233

Profesor îndrumător: Nandra Constantin

An universitar: 2021 – 2022

Cuprins

1. Prezentarea temei

Tema proiectului de față o reprezintă implementarea unei scene de obiecte 3D, folosind specificația OpenGL (Open Graphics Library)[1], care să dispună de un grad ridicat de fotorealism, realizat prin intermediul complexității, a detaliilor și a efectelor speciale ce definesc scena de obiecte.

Astfel, proiectul își propune redarea cât mai fidelă a unei scene reale, în care detaliile specifice cadrului 3D modelat să respecte principiile vizuale și estetice ale unei scene reale, incluzând elemente specifice realității, prin efecte precum ceață, umbre, fluctuațiile de nivel ale unui curs de apă, animații particulare, dar și obiecte statice care să se apropie de obiectele din lumea reală. Așadar, s-a ales spre a fi modelată o scenă din natură, surprinzând un mic sat de munte, cu case răsfirate, înconjurat de pădure și traversat de o apă curgătoare.

Scopul final al aplicației îl reprezintă posibilitatea vizualizării pe ecran și a manipulării de către utilizator a scenei modelate, folosind tastatura și mouse-ul calculatorului.

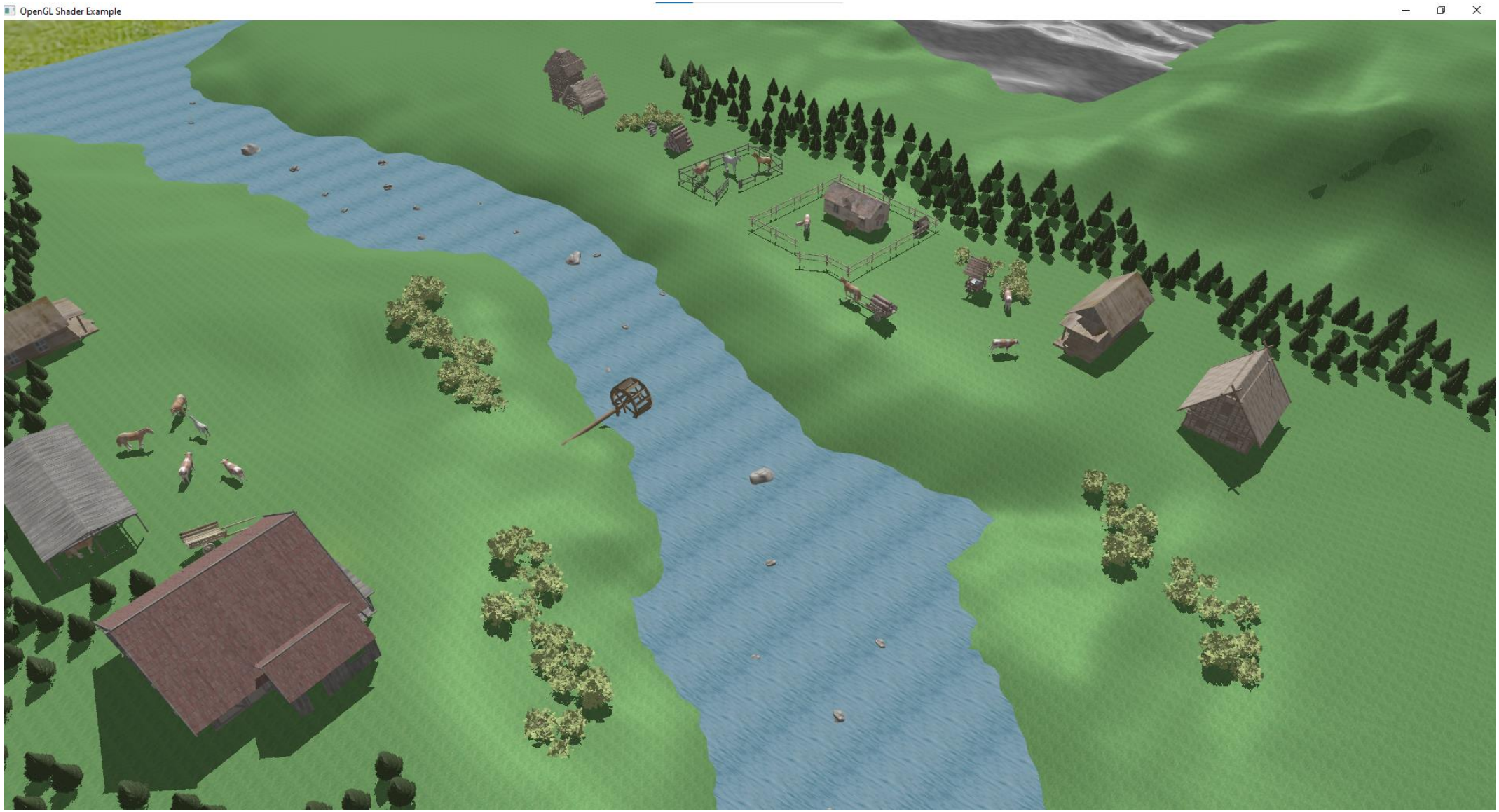
2. Scenariul

2.1. Descrierea scenei și a obiectelor

Scena modelată redă un peisaj montan, ce surprinde creste montane, dealuri și având în centru un sat mic de munte, traversat de un râu. Satul este format din șase case de lemn, unele având o gospodărie proprie. În scenă sunt prezente și animale domestice, fie aflate în adăposturi special amenajate, fie îngrădite. De asemenea, satul modelat este înconjurat de o pădure deasă de conifere, iar pe cursul râului se află o roată care prezintă o mișcare de rotație datorată curgerii apei.

Terenul modelat este unui denivelat, surprinzând forme de relief diverse: munți, dealuri, văi, pajiști, fotorealismul reprezentat prin diferențele de nivel dintre aceste forme de relief fiind totodată susținut și de prezența ceții. De asemenea, în scenă se observă umbrele obiectelor, formate datorită prezenței soarelui. Casele din componența satului sunt caracterizate de detalii specifice, atât în ceea ce privește culoarea, aspectul lemnului din care sunt construite, cât și conformația și planul construcției.

Scena mai sus descrisă poate fi vizualizată, în ansamblu, în figura de pe pagina următoare.



Figură 1 Viziune de ansamblu asuprea scenei

2.2. *Funcționalități*

Funcționalitățile principale ale aplicației sunt reprezentate de posibilitatea navigării libere în scena de obiecte modelată, prin folosirea mouse-ului și a tastaturii, pentru a putea vizualiza în detaliu fiecare obiect 3D prezent în scenă, dar și pentru o mai bună identificare a detaliilor, animațiilor și efectelor 3D ce conturează fotorealismul scenei. În scenă sunt prezente două surse de lumină diferite, una direcțională și una punctiformă, situată pe lampa din fața unei case.

De asemenea, anumite funcționalități ale aplicației pot fi controlate de către utilizator, prin apăsarea unor taste specifice, prezentate în cadrul capitolului 4, precum: posibilitatea rotirii sursei de lumină, pentru o mai bună vizualizare a efectelor de umbră și simularea mișcării soarelui pe cer în decursul unei zile, adăugarea sau eliminarea efectului de ceață, vizualizarea în mod zi și mod noapte, și vizionarea unui tur virtual (animație de prezentare) a întregii scene de obiecte.

Mai mult decât atât, prin apăsarea anumitor taste, utilizatorul are posibilitatea de a vizualiza și anumite detalii de implementare, cum ar modurile de vizualizare solid, point și wireframe[2], care surprind și elementele geometrice folosite pentru modelarea obiectelor, dar și pozițiile exacte ale acestora.

3. *Detalii de implementare*

3.1. *Funcții și algoritmi*

Implementarea aplicației a avut drept punct de pornire șablonul de cod folosit în cadrul laboratoarelor de pe parcursul semestrului la materia Prelucrare Grafică, motiv pentru care funcțiile principale și algoritmii implementați anterior au fost adaptați și utilizați în procesul de implementare a aplicației de față.

Principalele funcții utilizate, ale căror nume sunt sugestive pentru funcționalitățile pe care le implementează, sunt:

```
void windowResizeCallback(GLFWwindow* window, int width, int height)

void keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode)

void mouseCallback(GLFWwindow* window, double xpos, double ypos)

void processMovement()

bool initOpenGLWindow()

void initObjects()

void initShaders()

void initUniforms()

void drawObjects(gps::Shader shader, bool depthPass)

void renderScene()

void initSkyBox()

void cleanup()
```

Mișcarea camerei

Suplimentar funcțiilor anterior descrise, pentru realizarea funcționalității de navigare în scenă, au fost implementate operațiile corespunzătoare deplasării pe cele patru direcții posibile (înainte, înapoi, stânga, dreapta) și rotației în clasa `Camera`, având următorul antet de metodă:

```
//mișcarea camerei
void move(MOVE_DIRECTION direction, float speed);

//rotația camerei în jurul axelor x și y
void rotate(float pitch, float yaw);
```

Metodele mai sus menționate sunt apelate la înregistrarea unor evenimente specifice de la mouse și tastatură, tratate în cadrul funcțiilor:

```
void mouseCallback(GLFWwindow* window, double xpos, double ypos)
void processMovement()
```

Efectul de ceață

Pentru realizarea efectului de ceață, a fost implementată la nivelul shader-ului de fragmente principal următoarea funcție, care efectuează calculele necesare realizării unui efect de ceață, folosind varianta exponențială pătratică a acesteia, prezentată în cadrul laboratorului:

```
float computeFog()
```

Efectul dinamic al râului (valuri)

Pentru a crea un efect dinamic, care să redea mișcarea cursului de apă al râului, a fost necesară implementarea unei funcții de generare a unor perturbații random, care, prin poziționarea vârfurilor subdiviziunilor planului ce descrie râul, să dea iluzia unor valuri. Această funcție a fost preluată de la sursa [3], și implementată la nivelul shader-ului pentru vârfuri, folosit la desenarea cursului de apă:

```
// 2D Random
float random (in vec2 st) {
      return fract(sin(dot(st.xy, vec2(12.9898,78.233))) *
      inc);
}

// 2D Noise based on Morgan McGuire @morgan3d
// https://www.shadertoy.com/view/4dS3Wd
float noise (in vec2 st) {
      vec2 i = floor(st);
      vec2 f = fract(st);

      // Four corners in 2D of a tile
      float a = random(i);
      float b = random(i + vec2(1.0, 0.0));
      float c = random(i + vec2(0.0, 1.0));
      float d = random(i + vec2(1.0, 1.0));
```

```
        // Smooth Interpolation

        // Cubic Hermine Curve. Same as SmoothStep()
        vec2 u = f*f*(3.0-2.0*f);
        // u = smoothstep(0.,1.,f);

        // Mix 4 coorners percentages
        return mix(a, b, u.x) + (c - a)* u.y * (1.0 - u.x) + (d -
        b) * u.x * u.y;
}
```

Pentru obținerea perturbațiilor dorite, fiecare vârf al subdiviziunii planului folosit pentru descrierea râului a înregistrat o deplasare random pe axa Oy:

```
gl_Position = projection * view * model * vec4(vPosition.x,
noise(vPosition.xz), vPosition.z, 1.0f);
```

Efectele de lumină și umbră

Pentru obținerea efectului de lumină a fost folosit modelul de iluminare Phong, prezentat în laborator și implementat în cadrul shader-ului de fragmente, folosind funcția:

```
void computeLightComponents()
```

Pentru generarea umbrelor a fost folosită tehnica Shadow Mapping, implementarea funcției de calcul a umbrelor realizându-se, de asemenea, la nivelul shader-ului de fragmente:

```
float computeShadow()
```

3.1.1. Soluții posibile

Soluțiile mai sus prezentate și implementate în cadrul proiectului au, bineînțeles, metode alternative de realizare a unor efecte similare: pentru iluminarea globală s-ar fi putut folosi fie modelul Gourand, fie modelul Blinn-Phong, pentru realizaea efectului de ceață există alte două alternative de generare, ceața liniară și ceața exponențială, iar pentru obținerea efectului de mișcare a apei s-ar fi putut folosi translații spre poziții aleatoare, la momente de timp aleatoare

3.1.2. Motivarea abordării alese

Soluțiile și abordările adaptate în implementare au fost alese din motive de simplitate și claritate a codului, dar și din motive de eficiență și o mai bună stăpânire de către dezvoltatorul proiectului a noțiunilor și metodelor de prelucrare și manipulare folosite.

3.2. *Modelul grafic*

Modelul grafic al scenei a fost realizat folosind aplicația Blender, în care au fost importate obiecte 3D (de tip .obj), majoritatea descărcate de pe internet [4], [5], [6] și texturate, scalate, rotite și translatate mai apoi astfel încât să compună ansamblul scenei 3D.

Modelul terenului a fost preluat din setul disponibil în aplicația Blender și a fost ajustat și adaptat la cerințele aplicației. Pentru modelarea munților a fost folosit un template predefinit în Blender [7], iar textura aplicată este una diferită față de cea folosită la reprezentarea dealurilor, pentru a indica diferența de altitudine.

Scena conținând obiectele statice a fost modelată în întregime în Blender și exportată mai apoi ca un singur obiect 3D, în timp ce componentele mobile au fost prelucrare și exportate individual pentru a putea permite realizarea efectelor și a animațiilor din cadrul aplicației OpenGL prin prelucrarea directă a acestora și supunerea lor la o serie întreagă de transformări, în vederea obținerii unui nivel ridicat de fotorealism. Pentru desenarea cursului de apă au fost definite două shadere individuale, de vârfuri și de fragmente, care permit un control mai bun asupra animării subdiviziunilor componente ale acestuia, în tim ce restul obiectelor dinamice au fost desenate folosind shader-ul principal, folosit pentru întreaga scenă de obiecte, fără a necesita shadere individuale.

Sursa de lumină prezentă este una direcțională, permițând generarea umbrelor folosind tehnica Shadow Mapping și vizualizarea dinamică a acestora prin rotirea în jurul axei Oy.

Mediul înconjurător, nemodelat, este reprezentat de un SkyBox care completează peisajul modelat, conferindu-i un grad și mai înalt de fotorealism.

3.3. *Structuri de date*

Structurile de date folosite sunt cele furnizate în specificația OpenGL prezente în template-ul furnizat în cadrul laboratorului, nefiind definite noi structuri de date.

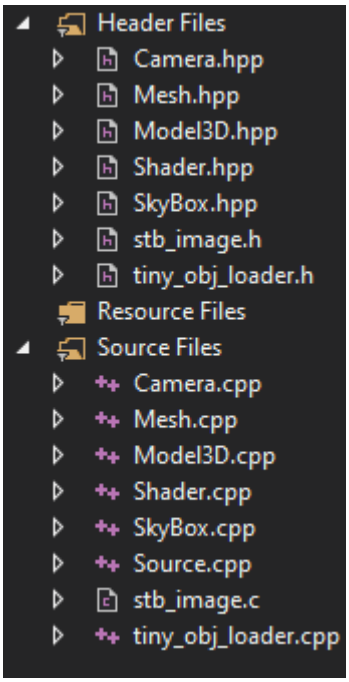
Printre structurile de date folosite se numără:

```
struct Vertex

{
	glm::vec3 Position;
	glm::vec3 Normal;
	glm::vec2 TexCoords;
};

struct Texture
{
	GLuint id;
	//ambientTexture, diffuseTexture, specularTexture
	std::string type;
	std::string path;
};

struct Material
{
	glm::vec3 ambient;
	glm::vec3 diffuse;
	glm::vec3 specular;
};
```

```
struct Buffers {
	GLuint VAO;
	GLuint VBO;
	GLuint EBO;
};
```

3.4. *Ierarhia de clase*

Clasele folosite la implementarea aplicației sunt cele predefinite în cadrul șablonului de cod furnizat, nefiind necesară definirea unor clase noi, ci doar adaptarea și completarea celor existente. Astfel, principalele clase folosite în implementarea proiectului, definite prin fișiere header (.hpp) și fișiere sursă C++ (.cpp), sunt vizibile în figura de mai jos:



Figură 2 Clasele folosite în implementarea aplicației

4. *Prezentarea interfeței grafice utilizator. Manual de utilizare*

Interfața grafică cu utilizatorul permite vizualizarea și navigarea în scena de obiecte modelată, folosind mouse-ul și tastatura.

Mouse-ul este utilizat pentru rotația camerei și pentru selectarea direcției și a volumului de vizualizare, intrările de la tastatură având următoarele semnificații:

- O – vizualizare scenă în modul Solid;
- P – vizualizare scenă în modul Point;
- I – vizualizare scenă în modul Wireframe;
- J, L – rotația sursei de lumină față de axa Oy;
- W – deplasarea camerei în față;

- A – deplasarea camerei la stânga;
- S – deplasarea camerei în spate;
- D – deplasarea camerei la dreapta;
- F, G – rotirea camerei în jurul axei Ox;
- V, B – rotirea camerei în jurul axei Oy;
- Y – vizualizarea scenei în prezența ceții;
- U – vizualizarea scenei în absența ceții;
- Z – turul virtual al scenei;
- N – vizualizare pe timp de noapte;
- X – vizualizare pe timp de ziuă.

5. *Concluzii și dezvoltări ulterioare*

În concluzie, se poate afirma că specificația OpenGL, dar software-ul de dezvoltare 3D, Blender, reprezintă unelte avansate de modelare și implementare a scenelor de obiecte 3D, găsindu-și o aplicabilitate largă în domeniul graficii computerizate în sfere de activitate precum jocuri pe calculator și animații. Pentru dezvoltatorul proiectului, implementarea acestuia a fost utilă, având drept rezultat o mai bună însușire a metodelor de modelare a obiectelor 3D, dar și de construire a aplicațiilor grafice folosind pipline-ul grafic pus la dispoziție de specificația OpenGL.

Printre posibilitățile de dezvoltare ulterioară a aplicației se numără introducerea unui avatar cu care utilizatorul să poată naviga în scena de obiecte, detecția coliziunilor astfel încât să nu fie permisă trecerea prin interiorul obiectelor, dar și introducerea de noi animații, astfel încât fotorealismul scenei să fie adus la un nivel superior.

6. *Referințe*

[1] OpenGL Reference Manual, Disponibil online: https://www.glprogramming.com/blue/

[2] OpenGL Reference Pages – glPolygonMode, Disponibil online: https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glPolygonMode.xhtml

[3] 1D, 2D & 3D Value Noise, Disponibil online: https://www.shadertoy.com/view/4dS3Wd

[4] CGTrader, Disponibil online: https://www.cgtrader.com/

[5] Free3D, Disponibil online: https://free3d.com/3d-models/

[6] TurboSquit, Disponibil online: https://www.turbosquid.com/

[7] Blender Quick Tip 003: Quick And Easy 3D Landscapes In Blender 2.8, Disponibil online: https://www.youtube.com/watch?v=IZXDDrXjwMA&t=56s