OpenGL Project

Prelucrare Grafica

Student: Bicioi Constantina Luiza

Grupa: 30238

Cuprins

1. Prezentarea temei 3
2. Scenariu 3
2.1 Descrierea scenei si a obiectelor 3
2.2 Functionalitati 5
3. Detalii de implementare 7
3.1 Functii si algoritmi 7
3.1.1 Functii si algoritmi 7
3.1.1 .1 Functii si algoritmi 7
3.1.1 .2 Motivatia abordarii alese 8
3.2 Structuri de date folosite 8
3.3 Ierarhia de clase 9
4. Manual de utilizare 9
5. Dezvoltari ulteriora 10
6. Concluzii 10

1. Prezentarea temei

Scopul proiectului este acela de a simula navigarea print-o scena complexa .

Proiectul meu se concentrează pe crearea unei scene principale în realitatea virtuală, inspirată de universul fascinant al pisicilor. În această lume virtuală, utilizatorii vor avea ocazia de a explora și interacționa cu personaje adorabile și peisaje fermecătoare, toate într-un mod captivant

Utilizatorul va identifica taramul pisicilor sau asa cum l-a numit colega mea de camera „Pisi Land”. In scena sunt prezente si animatii precum miscarea rotativa a unei rate pe apa, miscarea de translatie a baloanelor si miscarea efectiva a camerei.

De asemenea, utilizatorul poate sa interactioneze cu aplicatia prin intermediul mouse-ului si a tastaturii pentru miscarea in camera, deplasarea sau schimbarea sursei de lumina, cresterea intensitatii cetii etc.

2. Scenariu

Se porneste fisierul .exe , se prezinta scena intr-o scurta prezentare de camera (ma folosesc in principal de functia move pentru a putea vedea mai multe puncte ale scenei mele. (exemplu : myCamera.move(gps::MOVE_BACKWARD, animationSpeed * elapsedTime * 0.2); , una din miscariile animatiei de camera)

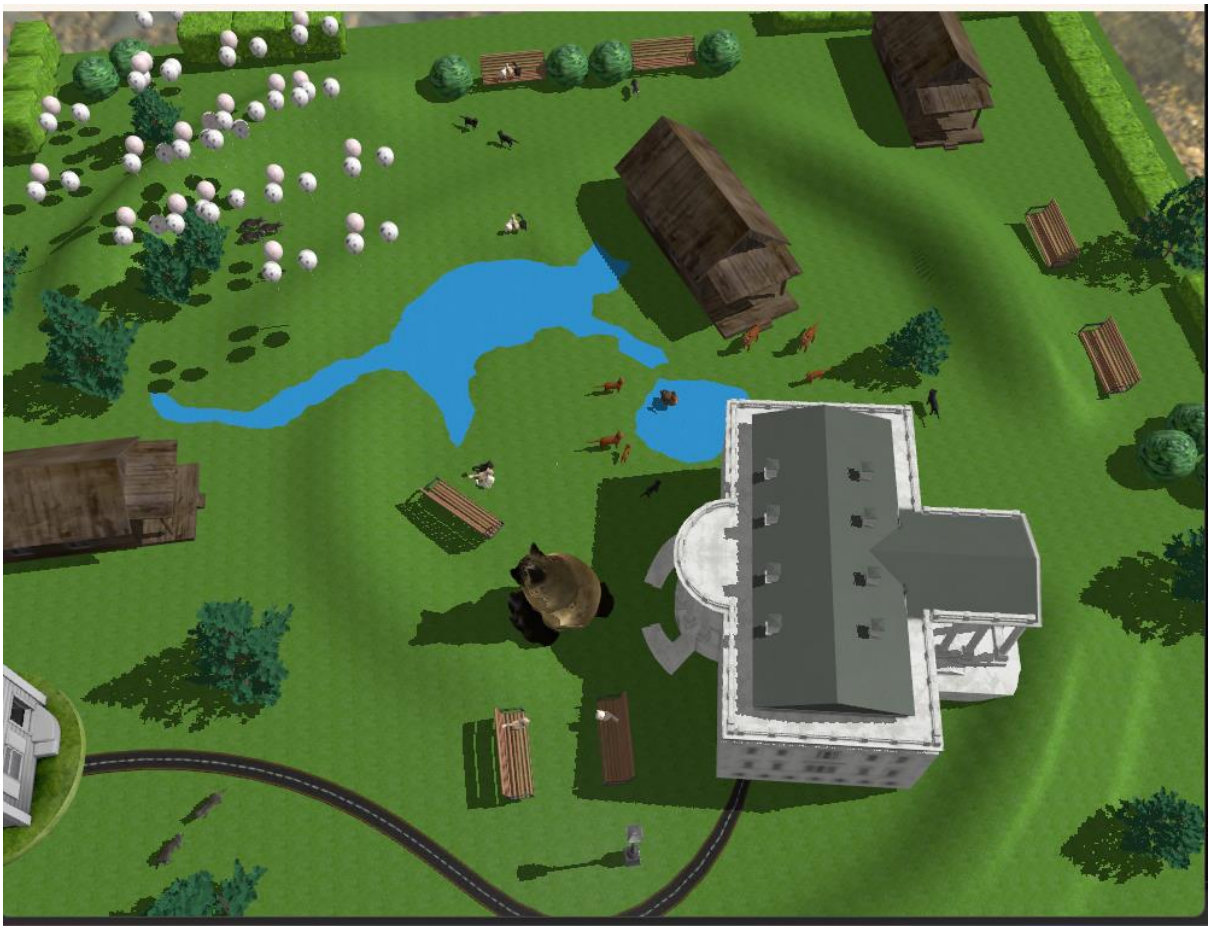
2.1 Descrierea scenei si a obiectelor

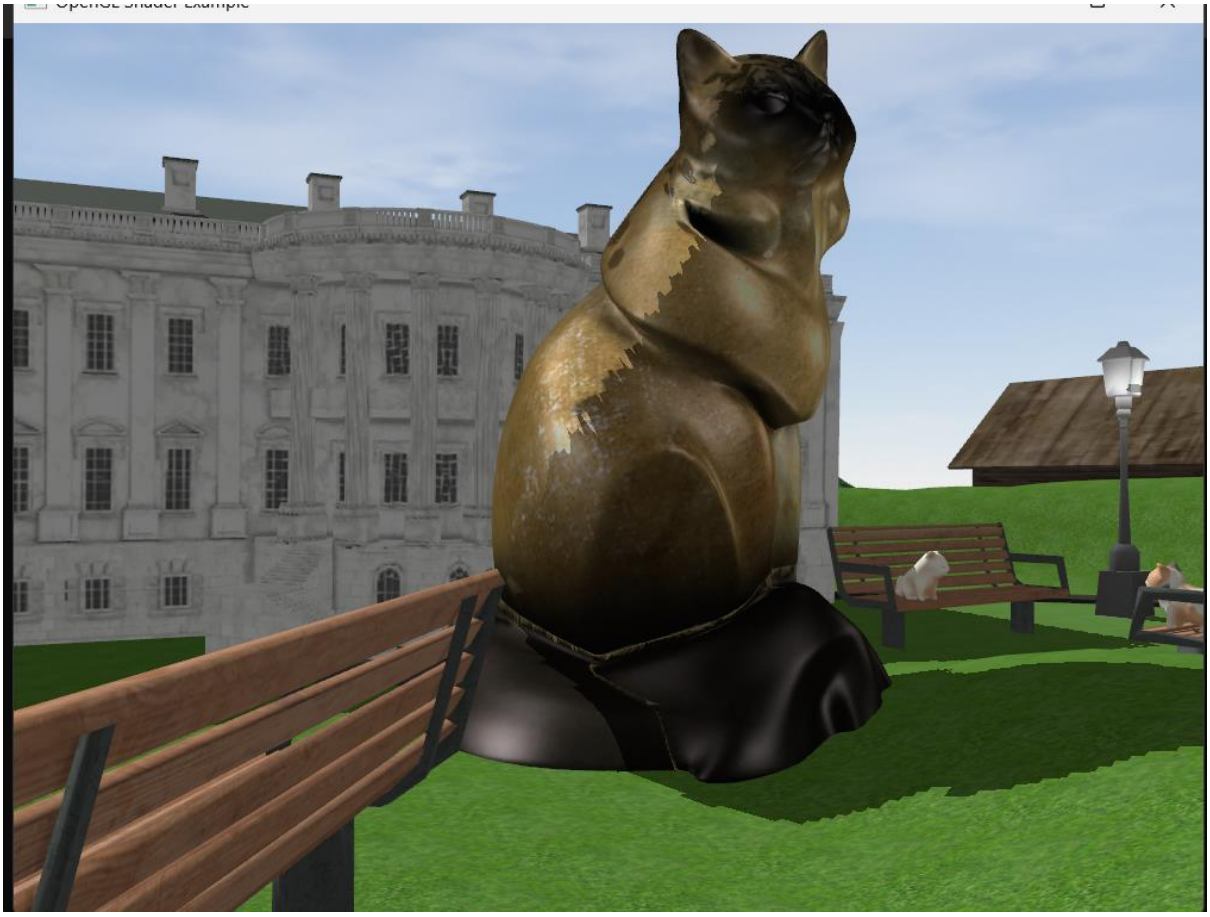
Scena aplicatiei se concentreaza pe un singur plan compact , usor de parcurs , avand continuitate si o logica in spate :

Deoarece am folosit template ul dat la laboratoare si nu project core-ul nu pot spune ca elemente create cu shader-ul basic.vert , basic.frag ci cu **shaderStart.vert** si **shaderStart.frag** :

- Cladiri de diverse forme si texturi
- Drumul facut prin aboradarea domnului profesor Nandra
- Pisici de diverese tipuri si marimi si de asemenea ratoni
- Stalp de iluminat si bancute
- Pomi si tot ce tine de natura

Dupa cum se observa in imaginile de mai jos, prin acest shader se obtin umbre ale obiectelor in functie de pozitia lor fata de lumina cu ajutorul functiei computeShadow() prezentata in laboratorul 9.





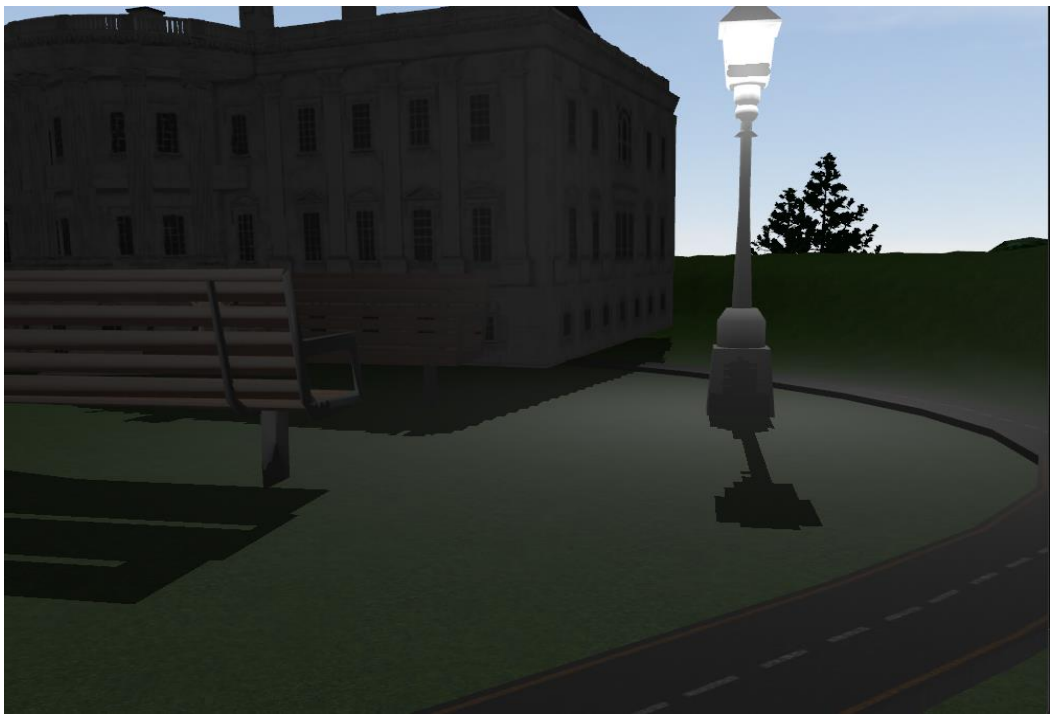
2.2 Functionalitati

- Vizualizarea scenei prin mouse si tastatura si a animatiei de prezentare (care incepe chiar la pornirea executabilului)
- Prezenta a doua surse de lumina distincte (directionala si punctiforma)
    - Se observa prezenta luminii directionale apasand tastele J si L (sau mai bine zis se observa schimbarea unghiului acestei lumini) , iar ajustarea sa (adica trecerea de la noapte la zi , se poate face prin tastele 5 si 6)
    - Lumina punctiforma este existenta , functia sa existand tot in shaderStart.frag - computeSpotLightComponents() -lab7
- Prezenta umbrelor in aplicatie

- Animatii : rotatie in jurul unui punct diferit de origine (centrul baltii micute de langa white house a pisicilor)
- Elementul de ceata -laboratorul 11
    - Pentru a face ceata tasta F
    - Pentru a ridica ceata tasta G
    - Functia in care a fost implementata : computeFog()

      $fogFactor = e^{-(fragmentDistance * fogDensity)}$ 2

- Obiectul luminat cu lumina punctiforma : lampa stradala
    - Activare/Dezactivare lumina: 7, 8



- Skybox – din laboratorul 10

3. Detalii de implementare

3.1 Functii si algoritmi

3.1.1 Functii si algoritmi

- `keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode)`
  - folosita pentru a receptiona butoanele la o singura apasare a lor
- `mouseCallback(GLFWwindow* window, double xpos, double ypos)`
  - folosita la miscarea in scena folosind mouse-ul
- `processMovement()`
  - folosita la receptionarea tastelor si (in cazul miscarii in scena), trimise mai departe la Camera
- `initObjects()`
- `initShaders()`
- `initSkyBox()`
- `initFBO()`
  - folosita la initializarea hartii de adancime a scenei
- `drawObjects(gps::Shader shader, bool depthPass)`
  - Functie folosita la desenarea obiectelor pentru care vreau sa am umbra
- `renderScene()`
  - functie folosita la randerizarea obiectelor
- `initAnimation() si updateCameraAnimation()`
  - functie cu ajutorul careia scena se plimba singura in camera

3.1.1 .1 Functii si algoritmi

- **Algoritmul Shadow Mapping -laborator 9**
  - Tehnica multi-trecere ce utilizeaza texturi de adancime pentru a decide daca un obiect se afla sau nu in umbra
  - Sursa de lumina e considerata locatia camerei; daca un obiect din scena nu poate fi vazut din perspectiva camerei, atunci el se afla in umbra
  - Etape
    Etapele principale ale algoritmului sunt descrise mai jos::

1. Rasterizarea scenei din punctul de vedere al luminii. Nu contează cum arată scena (informații
despre culoare); singurele informații relevante în acest moment sunt valorile de adâncime. Aceste
valori sunt stocate într-o hartă de umbră (sau hartă de adâncime) și pot fi obținute prin crearea
unei texturi de adâncime, atașarea la un obiect framebuffer și rasterizarea întregii scene (așa cum
este văzută din poziția luminii) în acest obiect. În acest fel, textura de adâncime este umplută direct
cu valorile relevante ale adâncimii.
2. Rasterizarea scenei din punct de vedere al observatorului (poziția camerei). Se compară
adâncimea fiecărui fragment vizibil (proiectat în cadrul de referință al luminii) cu valorile de
adâncime din harta umbrelor. Fragmentele care au o adâncime mai mare decât cea care a fost
stocată anterior în harta de adâncime nu sunt direct vizibile din punctul de vedere al luminii și
sunt, prin urmare, în umbră.

- Algoritmii de animatie
  - Se bazeaza pe rotatii in jurul unui punct diferit de origine, deci e nevoie de translatia noului punct in punctul de origine, urmata de rotatie cu numarul de grade dorit, si ulterior translatia inapoi in punctul initial.
  - Se bazeaza pe translatia in functie de delta a obiectului in cauza (*inspiratie laborator 6*)

3.1.1 .2 Motivatia abordarii alese

Generarea de umbre cu Shadow Mapping: mai simplu de implementat si integrat in aplicetie.

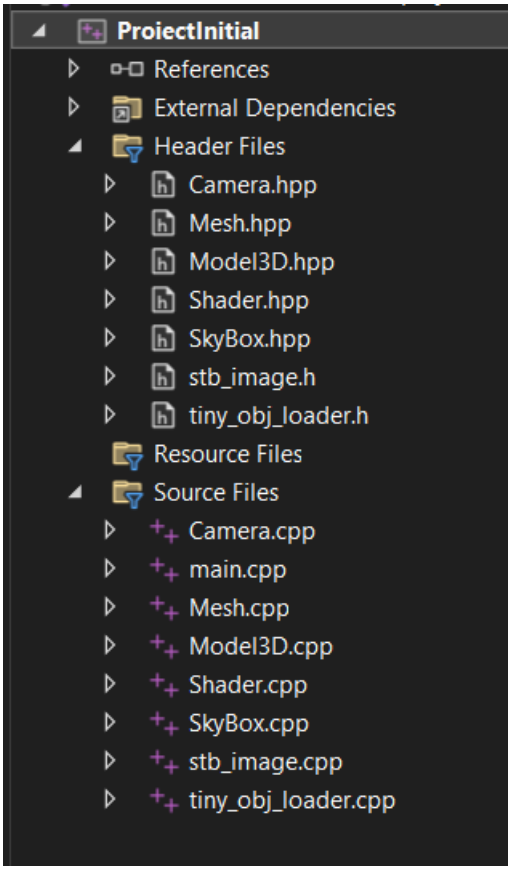
Animatia de rotatie in jurul unui punct diferit de origine implica un grad de complexitate mai mare si confera realism scenei , in timp ce **miscare de translatie** in functie de un delta care tine si de timpul de pornire a aplicatiei confera o urmarire mai lina a miscarii baloanelor . Dupa un anumit timp , ele se vor intoarce in pozitia initiala si isi vor relua traseul de „zbor” in urma calcularii aceluiasi algoritm

3.2 Structuri de date folosite

In cadrul proiectului nu s-au definit noi structuri de date customizate, ci s-au folosit deja cele existente precum:

- Valori primitive: Gluint
- Matrice si vectori: glm::mat2, glm::vec3;

3.3 Ierarhia de clase



4. Manual de utilizare

- Mouse: miscari de rotatie in camera
- W, S, D, S – sus, jos, in spate
- Q, E – rotatie stanga, dreapta in jurul axei Y
- 0- afisarea SOLID a scenei
- 1 – afisarea WIREFRAME
- 2 – afisarea POLIGONAL
- 3- afisarea SMOOTH
- F,G – modificare intensitate ceata
- J,L – rotirea sursei de lumina directionala (+-1.0f)

5. Dezvoltari ulteriora

O posibila si seminificativa imbunatatire a programului este aceea de adaugare a ploii sau a fulgerului. De asemenea, aplicatia poate fi imbunatatita si prin crearea unor animatii mai complexe (miscarea pisicilor din scena spre exemplu sau adaugarea unei masini care sa urmareasca drumul serpuit).

6. Concluzii

Elaborarea scenei complexe in OpenGL, ce ilustreaza o lume a pisicilor (la care am adaugat pisicile interzise adica ratonii), reprezinta o experienta semnificativa in formarea mea . În concluzie, procesul complex de proiectare și dezvoltare al scenei "Tărâmul Pisicilor" a reprezentat o provocare captivantă, determinându-mă să explorez și să aplic concepte avansate în OpenGL. Prin această experiență, am acumulat cunoștințe semnificative în gestionarea eficientă a resurselor grafice, asigurându-mă că performanțele sunt optime pentru a oferi o experiență fluentă și captivantă utilizatorilor.