UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

# Proiect

*PRELUCRARE GRAFICA*

Nume si prenume: Halmaciu Delia

Grupa: 30236

Facultatea de Automatica si Calculatoare

12 Ianuarie 2025

# Cuprins

# 1 Prezentarea temei

Proiectul constă într-o aplicație OpenGL ce oferă o scenă 3D interactivă, creată pentru a demonstra tehnici moderne de grafică computerizată. Scena integrează mai multe modele 3D, inclusiv o plaja, valuri dinamice, felinare și un skybox pentru fundal, toate fiind proiectate să creeze o atmosferă realistă și captivantă. Utilizatorul are posibilitatea de a naviga liber prin scenă folosind o cameră controlată prin input-uri de la tastatură și mouse, având opțiunea de a modifica dimensiunea, poziția și rotația obiectelor.

# 2 Scenariu

## 2.1 Descrierea scenei și a obiectelor

Scena reprezinta o plaja cu o casa moderna,fantana si statui,palmieri,2 sezlonguri ,umbrela de plaja,minge ,un leagan,un castel de nisip si vedere catre mare unde se pot observa niste delfini.

Scena este formata din mai multe obiecte 3D:

- 2 plane:unul pentru nisip,unul pentru mare
- casa moderna
- palmierii
- fantana si statuile
- mingea de plaja
- umbrela de plaja
- 2 sezlonguri
- valurile din mare
- cei 3 delfini
- castelul de nidip
- leahanul
- 2 felinare

Tuturor acestor obiecte le sunt atribuite texturi corespunzatoare si realiste.

Valurile din apa sunt animate pentru a oferi un efect realist al marii.

Figura 1: Plaja

### 2.2 Funcționalități

1. Miscarea camerei
   Utilizatorul poate explora scena într-un mod interactiv prin mișcarea camerei în toate direcțiile: înainte, înapoi, la stânga, la dreapta, în sus și în jos, folosind tastele dedicate. În plus, unghiul de vizualizare poate fi ajustat cu ușurință prin mișcarea mouse-ului, oferind o experiență flexibilă și intuitivă de navigare.

2. Animatile valurilor
   Adaugă un nivel suplimentar de interactivitate si realism scenei.

## 3 Detalii de implementare

### 3.1 Funcții și algortmi

### 3.2 Soluții posibile

Aplicația utilizează mai multe tehnici de grafică 3D:

- Shader-e: Shader-urile sunt utilizate pentru a controla modul în care obiectele sunt renderizate pe ecran. Acestea includ shader-e pentru iluminare, efecte de umbre și texturi.
  . Soluție: Am utilizat shader-e scrise în limbajul GLSL pentru a implementa iluminarea, umbrele și texturile. Shader-ul de iluminare folosește un model de iluminare Phong pentru a oferi un aspect realist, combinând componentele ambientale, difuze și speculare. Texturile sunt aplicate prin intermediul coordonatelor UV, iar efectele de umbre sunt generate folosind o hartă de umbre (shadow mapping).

- Cameră: Mișcarea camerei este gestionată printr-un sistem de control bazat pe taste și mișcarea mouse-ului.
  . Soluție: Sistemul de mișcare a camerei este bazat pe manipularea matricelor de proiecție și vizualizare. Mișcările translaționale (față, spate, stânga, dreapta, sus, jos) sunt gestionate prin combinarea tastelor WASD și săgeților, iar rotația camerei este controlată prin poziția mouse-ului utilizând un mecanism de "look-around".

- Animatii
  . Soluție: Animațiile sunt implementate prin interpolare liniară și sinusoidală, folosite pentru a genera mișcări naturale, precum valurile apei sau mișcarea obiectelor dinamice din scenă. S-a utilizat un timer bazat pe un shader sau pe sincronizarea cu ceasul aplicației pentru a garanta continuitatea mișcării.

- SkyBox
  . Soluție: Skybox-ul este realizat utilizând un cub map texturat, unde fiecare față a cubului reprezintă o secțiune a panoramei cerului și a mediului înconjurător. Texturile cub map sunt încărcate dintr-un set de imagini HDR pentru a oferi un aspect imersiv.

### 3.3 Motivarea abordării alese

- Motivație Shader-e: Această abordare permite o personalizare avansată a aspectului grafic și o performanță optimă, având în vedere controlul direct asupra fiecărui pas din pipeline-ul grafic.
- Motivație Camera: Această soluție oferă o navigare fluidă și intuitivă pentru utilizator, similară cu cele mai populare sisteme de control din jocurile video, ceea ce sporește interactivitatea și confortul utilizării.
- Motivație Animatii: Această abordare simplă dar eficientă permite o simulare realistă a mișcărilor, cu un impact redus asupra performanței, fiind ideală pentru efectele vizuale dinamice ale unei scene 3D interactive.
- Motivație SkyBox: Alegerea cub map-urilor permite crearea unui fundal panoramic realist, care înconjoară scena în totalitate, îmbunătățind senzația de adâncime și realism fără a consuma resurse semnificative ale procesorului grafic.

### 3.4 Modelul grafic

Scena este formată dintr-o serie de obiecte 3D, fiecare reprezentând un model încărcat dintr-un fișier OBJ. Modelele sunt procesate folosind shader-e pentru a le aplica texturi și efecte de iluminare. Aceste obiecte sunt plasate într-un spațiu 3D definit de matricele de transformare.

1. Obiectele 3D

   Obiectele 3D din scenă, cum ar fi casa, palmierii, delfinii și fântâna, sunt încărcate din fișiere de tip .obj sau .fbx. Aceste fișiere conțin informații despre geometria și materialele obiectelor si au fost create in Blender.

   Texturile sunt aplicate prin intermediul coordonatelor UV, iar iluminarea este calculată folosind shader-e personalizate.

```
void initModels() {
    teapot.LoadModel("models/plaja/plajabuna.obj");
    std::cout << "Model loaded successfully!" << std::endl;

    wave2obj.LoadModel("models/plaja/wavebun.obj");
    std::cout << "New model 'wave2obj.obj' loaded successfully!" << std::endl;

    wave2obj.LoadModel("models/plaja/wavebun2.obj");
    std::cout << "New model 'wave2obj.obj' loaded successfully!" << std::endl;

    felinar1.LoadModel("models/plaja/felinar1.obj");
    std::cout << "New model 'wave2obj.obj' loaded successfully!" << std::endl;

    felinar2.LoadModel("models/plaja/felinar2.obj");
    std::cout << "New model 'wave2obj.obj' loaded successfully!" << std::endl;
}
```

## 2. Camera

Camera este implementată utilizând o matrice de vizualizare și o matrice de proiecție.Si prin taste se controleaza miscarea acesteia.

```
void processMovement() {
    if (pressedKeys[GLFW_KEY_W]) {
        myCamera.move(gps::MOVE_FORWARD, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_S]) {
        myCamera.move(gps::MOVE_BACKWARD, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_A]) {
        myCamera.move(gps::MOVE_LEFT, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_D]) {
        myCamera.move(gps::MOVE_RIGHT, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_UP]) {
        myCamera.move(gps::MOVE_UP, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_DOWN]) {
        myCamera.move(gps::MOVE_DOWN, cameraSpeed);
    }
    if (pressedKeys[GLFW_KEY_R]) {
        angle += 1.0f;
        model = glm::rotate(glm::mat4(1.0f), glm::radians(angle), glm::vec3(0.0f, 1.0f,
    }
    if (pressedKeys[GLFW_KEY_F]) {
        angle -= 1.0f;
        model = glm::rotate(glm::mat4(1.0f), glm::radians(angle), glm::vec3(0.0f, 1.0f,
    }
    if (pressedKeys[GLFW_KEY_Z]) {
        model = glm::scale(model, glm::vec3(0.95f, 0.95f, 0.95f));
    }
    if (pressedKeys[GLFW_KEY_X]) {
        model = glm::scale(model, glm::vec3(1.05f, 1.05f, 1.05f));
    }
    if (pressedKeys[GLFW_KEY_1]) {
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    }
    if (pressedKeys[GLFW_KEY_2]) {
        glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
    }
    if (pressedKeys[GLFW_KEY_3]) {
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    }
}
```

3. SkyBox

Skybox-ul este creat folosind un cub map texturat. Acesta constă în șase imagini (față, spate, sus, jos, stânga, dreapta) care sunt mapate pe fețele unui cub.Cubul este desenat în jurul camerei și este scalat suficient de mare pentru a da iluzia unui orizont infinit.

```
void initSkyBox() {
    std::vector<const GLchar*> faces;
    faces.push_back("skybox/posx.jpg");
    faces.push_back("skybox/negx.jpg");
    faces.push_back("skybox/posy.jpg");
    faces.push_back("skybox/negy.jpg");
    faces.push_back("skybox/posz.jpg");
    faces.push_back("skybox/negz.jpg");
    mySkyBox.Load(faces);
}
```

### 3.5 Structuri de date

Structurile de date sunt fundamentale pentru gestionarea obiectelor 3D, a camerei, a transformărilor geometrice, a mișcărilor și a interacțiunii utilizatorului cu scena. De exemplu : glm::mat4 și glm::mat3

Utilizate pentru matricele de transformare (model, vizualizare, proiecție) și pentru calculul normalMatrix.

```
    model = glm::rotate(glm::mat4(1.0f), glm::radians(angle), glm::vec3(0.0f, 1.0f, 0.0f));
    view = myCamera.getViewMatrix();
    projection = glm::perspective(glm::radians(45.0f),
      (float)myWindow.getWindowDimensions().width / (float)myWindow.getWindowDimensions().hei
```

### 3.6 Ierarhia de clase

Clasele utilizate:

1. Window: Clasa care se ocupă de crearea și gestionarea ferestrei de OpenGL.
2. Shader: Clasa care gestionează încărcarea și utilizarea shader-elor.
3. Camera: Clasa care controlează mișcarea camerei și vizualizarea scenei.
4. Model3D: Clasa pentru încărcarea și desenarea obiectelor 3D.
5. SkyBox: Clasa pentru gestionarea fundalului cerului.

## 4 Prezentarea interfeței grafice utilizator / manual de utilizare

### Comenzi disponibile:

1. Mișcare cameră:
   - W (înainte)
   - S (înapoi)
   - A (stânga)
   - D (dreapta)
   - UP (sus)
   - DOWN (jos).

2. Zoom: Folosește rotița mouse-ului.
3. Rotație obiecte:
    - R (rotire dreapta)
    - F (rotire stânga)
4. Moduri de vizualizare:
    - 1-wireframe
    - 2-polygonal
    - 3-solid
5. Mărire/micșorare obiecte:
    - Z (micșorare)
    - X (mărire)

# 5 Concluzii și dezvoltări ulterioare

În concluzie, proiectul reprezintă o aplicație interactivă OpenGL care combină tehnici avansate de grafică 3D pentru a crea o experiență captivantă și dinamică. Implementarea unei scene 3D detaliate, care include obiecte interactive, mișcarea camerei, animații și un skybox, permite utilizatorilor să exploreze și să interacționeze cu un mediu virtual complex. Prin utilizarea unor structuri de date eficiente și a unor algoritmi de redare și transformare, aplicația oferă performanță și o experiență vizuală de înaltă calitate. Abordările alese pentru fiecare componentă au contribuit la crearea unui sistem flexibil, ușor de extins și de personalizat, fiind o bază solidă pentru dezvoltări viitoare.

Pentru a extinde și îmbunătăți aplicația 3D OpenGL, iată câteva funcții pe care le-ai putea implementa în viitor:

- Implementarea unui sistem de selecție a obiectelor din scenă, utilizând mouse-ul, care să permită utilizatorului să aplice modificări asupra acestora (de exemplu, schimbarea culorii, rotație, scalare).
- Crearea unor animații care sunt declanșate de interacțiuni ale utilizatorului (ex: când un obiect este selectat sau mutat).
- Permite utilizatorului să interacționeze cu skybox-ul sau să modifice scena în funcție de mediul înconjurător (ex. schimbarea cerului sau a luminii pe măsură ce utilizatorul se deplasează).

# 6 Referințe


- OpenGL Documentation: https://www.opengl.org/documentation/
- GLM Documentation: https://github.com/g-truc/glm
- GLFW Documentation: https://www.glfw.org/documentation/