Sisteme de prelucrare grafica

Titlul proiectului: Pădure

Autor: Miklós Balázs

Grupa: 30234

Facultatea de automatica si calculatoare

UTCN

Cuprins

1. Descrierea cerintei

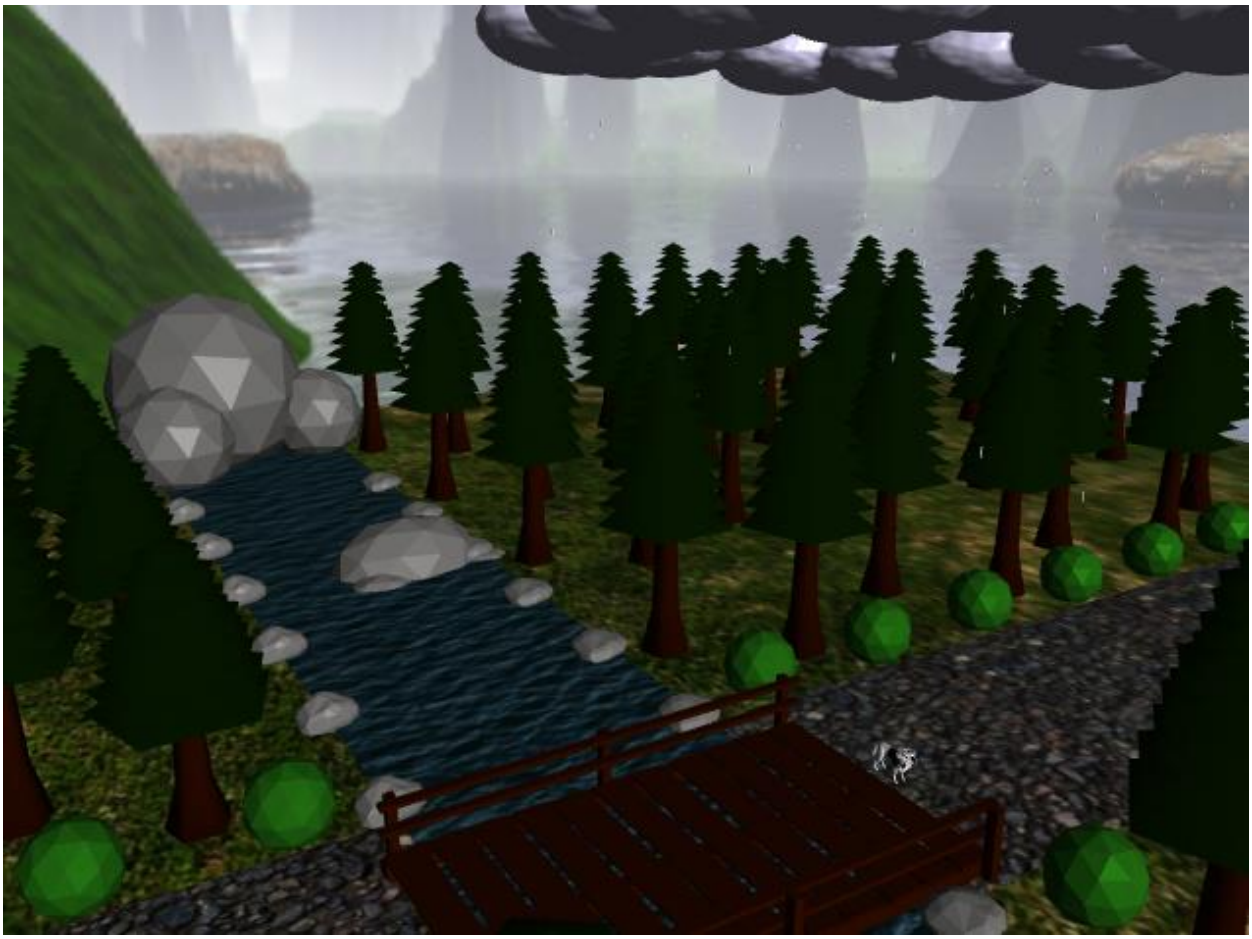
Proiectul are ca si scop realizarea unei prezentare fotorealistica a un scene de obiecte 3D utilizând librăriile prezentate la laborator (OpenGl, GLFW, GLM, etc.). Utilizatorul trebuie sa aiba posibilitatea de a controla scena prin intermediul mausului si tastaturii. In cazul proiectului meu scena pe care reprezint este o padure.

2. Descrierea scenei si a obiectelor, functionalitati

Fiindca tema proiectului este padurea, este evident ca aici folosesc obiecte pe care pot sa descriu mediul unei padure: copacii, diferite plante, animale, piatre. Ca obicte pentru animatia de ploaie folosesc un nor si cuboide mici ca fiind ei ploaia. Mai este desenata un rau animat si un drum prin care trece un pod. Caracterul este un lup, cu care putem și mișca.

Toata scena este desenata pe un obiect numit ground. (pe care tot am folosit si in laborator). In general am incercat ca sa folosesc obiecte pe care nu consuma prea mult procesorul si GPU- l calculatoruli, de aceea majoritatea obiectelor nu au atat de multe varfuri si au o textura simpla.

Scena arata asemanator ca si in figura de mai jos:



3. Detalii de implementare

a. Functii si algoritmi

Fiindca desenez mai multi obiecte din acelasi gen am avut nevoie de functii pe care generalizeaza desenarea obiectelor. Pentru obiectele pe care am desenat de mai multe ori am folosit functia creat de mine numit drawObject. Functia primeste ca parametru un tip Model3D, trei valori pentru translatarea si trei valori pentru scalarea obiectelor pe care le desenez in scena pe ground. Groundul tot este desenat cu ajutorul acestei functie, fiinca si pe el desenez de mai multe ori cu alt textura, cand este vorba despre drumul si raul.

Pentru obictele pe care am nevoie sa misc sau sa rotesc (pod si lup), am facut functie separata asemanator cu cea drawObject() doar mai rotesc pe ele sau misc dupa niste marametrii.

Pentru a desena mai muti obiecte pe ground matricial, foloseam doua cicluri de for si dupa variabilele for ului apelez functia draw object, obiectele sunt translatate dupa valoarea variabilelor. Dupa logica sau algoritmul aceasta desenezi mai multi tipuri de obiecte ca si copacii, piatre, plante. Și animatia de ploaie fac tot asa, doar fiindca toata functia renderScene() unde desenez, apare intr-un ciclu infinit, de aceea am venit cu o idee, cand desenez ploaia nu modific coordonatele x si z si modific coordonatele y random. Astfel arata ca si ploaie.

Pentru miscarea camerei mai sunt implmentate si functiile keyboardCallBack() si mouseCallback() pe care sunt apelate in functia renderScene() pana la terminarea programului.

b. Modelul grafic

[2]Modelul grafic in proiectul aceasta este o matrice de 4 dimensiuni. Cu ajutorul matricii desenez pe toata obiectele. Modelul este modificat in shadere vertex si fragment. De la vertex shader iese afara vectorul normal, positia ociului, positia LightSpace si positia cordonatelor de textua. Totusi in vertex shader mai acem si matricii projection si view si lightSpaceMatrix, cu care este inmultit matricea model ca sa iasa modelul final realistic. Iesirea este gl_Position, adica positia obiectelor in scena.

[2]Aceste elemente intra in frangment shader. In fragment shader outputul este culoarea obiectelor, care este facut dupa inputurile de la vertex shader. Culorile au trei componente, ambient, specular si difuse. Mai este facut su o functie compute shadow. Dupa combinarea celor trei componente cu umbrela, iese culoarea finala a modelelor texturat.

c. Structuri de date, ierarhia de clase

In proiectul aceasta sunt folosite mai multi structuri de date pentru a simplifica desenarea, miscarea sau colorarea obiectelor precum si a camerei de vizualizare. Aceste structuri de date au clase separate cu parametrii si functii separate pe care apelez pe parcursul desenarii.

[1]Pentru camera de vizualizare am o clasa numit Camera. La declararea camerei am doua vectori de trei dimensiuni, positia camerei, si unde se uita camera, sau cameraTarget. Dupa aceste vectori calculez directia de vizualizare (cameraDirection si cameraRightDirection). Mai sunt implementate niste functii getViewMatrix() penru a afla matricea de vizualizare a camerei si getCameraTarget() pentru a returna target-ul camerei. Pentru miscarea camerei dupa keyBoard este implementata functia move() si pentru rotirea camerei e implementata functia rotate().

[1]Mai avem clasele Mesh si Model3D pentru desenarea vertexurilor a obiectelor. Clasa Mesh este flosita diar in clasa Model3D pentru a modela varfurile obiectelor. In clasa Mesh avem trei structuri de date si anume Vertex cu elementele Position, Normal, si TextCoords pentru un varf a unui obiect. Mai avem structura Texture pentru textura unui obiect si structura Material cu componentele ambient, difuse si specular pentru culoarea obiectelor. Aceste structuri sunt folosite mai ales in shadere. Aici e definit si functia Draw() pentru desenarea obiectelor. De fapt in clasa Model3D functia Draw() foloseste metoda Draw() din clasa Mesh, desenand toata mash-urile adica varfurile la un obiect. Functiile care folosesc mai mut aici este functia pentru incarcarea obiectelor in program, adica constructorul si functia Draw() pentru desenarea obiectelor.

Mai avem o clasa Shader pentru shadere. In Clasa aceasta functiile pe care folosesc in proiectul de mai multe ori sunt loadShader pentru incarccarea shaderelor si useShaderProgram pentru a folosi un anumit shader la o parte din desenare a obiectelor. In afara de shadere principale vertex si fragment, mai avem doi shadere, un DepthMapShader, la care avem nevoie la desenarea umbleror, si un lightCubShader pentru a desena sursa de lumina. Este implementat doua sursa de lumina in proiectul aceasta si anume lumina alba pentru a simula ziua, si lumina albastru pentru a simula noaptea.

SkyBoxShader est folosit pentru a desena un SkyBox, adica un background in jurul scenei ca sa apare mai realistic. Pentru SkyBox avem o clasa numit SkyBox. In clasa aceasta folosesc functiile Load pentru a incarca cele sase parti a unui SkyBox, si mai folosesc functia Draw() pentru a desena SkyBox- ul. Shaderul folosit evident ca este SkyBoxShader aici.

4. Manual de utilizare

Dupa rularea programului, apare o fereastra unde este desenata scena cu toata obiectele.

Camera poate fi controlata folosind tastatura si mouse- ul. Pentru miscarea camerei inainte, inapoi la dreapta si la stanga trebuie sa apasam W, A, S si D pe tastatura. Pentru a roti camera, trebuie sa folsim mouse si sa miscam, unde vrem sa rotim camera si unde vrem sa vizualizam scena.

Pentru a porni animaia de ploaie trebuie sa apasam pe tastatura butonul I. Pentru a misca lupul folosim T (inainte), F (inapoi), G (la stanga) si H (la dreapta). Pentru a schimba intre surse de lumina sa folosim literele Q pentru noapte si E pentru zi.

Pentru a vizualiza scena in mod poligonal, punctual respectiv normal sa folosim tastaturule X(poligonal), C (punctual), si Z (normal). By default este in mod normal. Mod normal inseamna ca obiectelese sunt cu culori si cu texturi si nu se vede doar liniile sau varfurile obiectelor. Pentru a roti sursa de lumina puetem folosi tastele J si L. ATENTIE OPRIREA ANIMATIEI DE PREZENTARE se face cu litera Y.

5. Concluzii si dezvoltari ulterioare

Ca si concluzie pot sa afirm ca am invatat mult pe parcusul proiectului. Chiar daca nu a iesit perfect tot ce am vrut sa fac, am invatat cum functioneaza in mare parte openGl. Am invatat pentru ce sunt buni shaderele, cum sa incarcarm si sa desenam obiecte intr-o fereastra, cum sa desenam skyBox. In plus m-am dat seama cum sa folosim si texturile la obiecte si cum sa fac animatii.

Din cauza proiectului aceasta, m-am dat seama cat de multe lucru si effort trebuie ca un joc sa fie facut si sa si arata bine sau orice alt program cu interfata grafica mai complexa.

Exista mai multe dezvoltare al proiectului, fiindca gandirea umana daca este vorba despre creativitate este infinita. Aceasta inseamna ca mai putem sa adaugam obiecte la scena, mai putem sa marim hartia, scena. Tot asa si desenarea umbrelor la obiecte poate fi facut, texturarea mai detaliata, mai complexa, dar pentru aceasta tot asa trebuie incarcate obiecte cu mai multe detalii si varfuri ca sa putem aplica texturi mai detaliate si mai realistice pe ele. Tot asa, daca e cazul putem sa facem si un joc unde exista mai multi player, si exista mai mule enemy. Deci sunt mai multi cazuri dintre care putem alege ca si dezvoltare ulterioara.

6. Referinte

1. https://learnopengl.com/
2. https://moodle.cs.utcluj.ro/course/view.php?id=186