Prelucrare Grafica

~documentatie~

Student: Atitienei Stefan Costin
Grupa: 30237
Profesor de laborator: Constantin Ioan Nandra

Contents

1. Prezentarea temei

Tema acestui proiect este o scena medievala. Aceasta cuprinde un castel medieval, cu numeroase case de diferite forme, animale, ziduri si turnuri care inconjoara castelul si niste catapulte care apara castelul. Scena este plasata intre munti si dealuri, langa un lac pe care se afla niste corabii de pirati care ataca castelul. Acesta din urma poate arunca cu o ghiulea spre o barca si aceasta se va scufunda. O alta animatie prezenta in scena este elicea morii. O elice se invarte singura constant la o anumita viteza, iar o alta elice poate fi controlata prin taste. Tot prin taste, utilizatorul poate selecta diferite efecte, precum ceata, ploaia, poate seta ca scena sa aiba loc ziua sau noaptea, poate varia forma soarelui (ca lumina punctiforma sau directionala), poate misca soarele in cerc si tot de pe taste poate declansa un atac catre corabiile de pirati de pe lac.

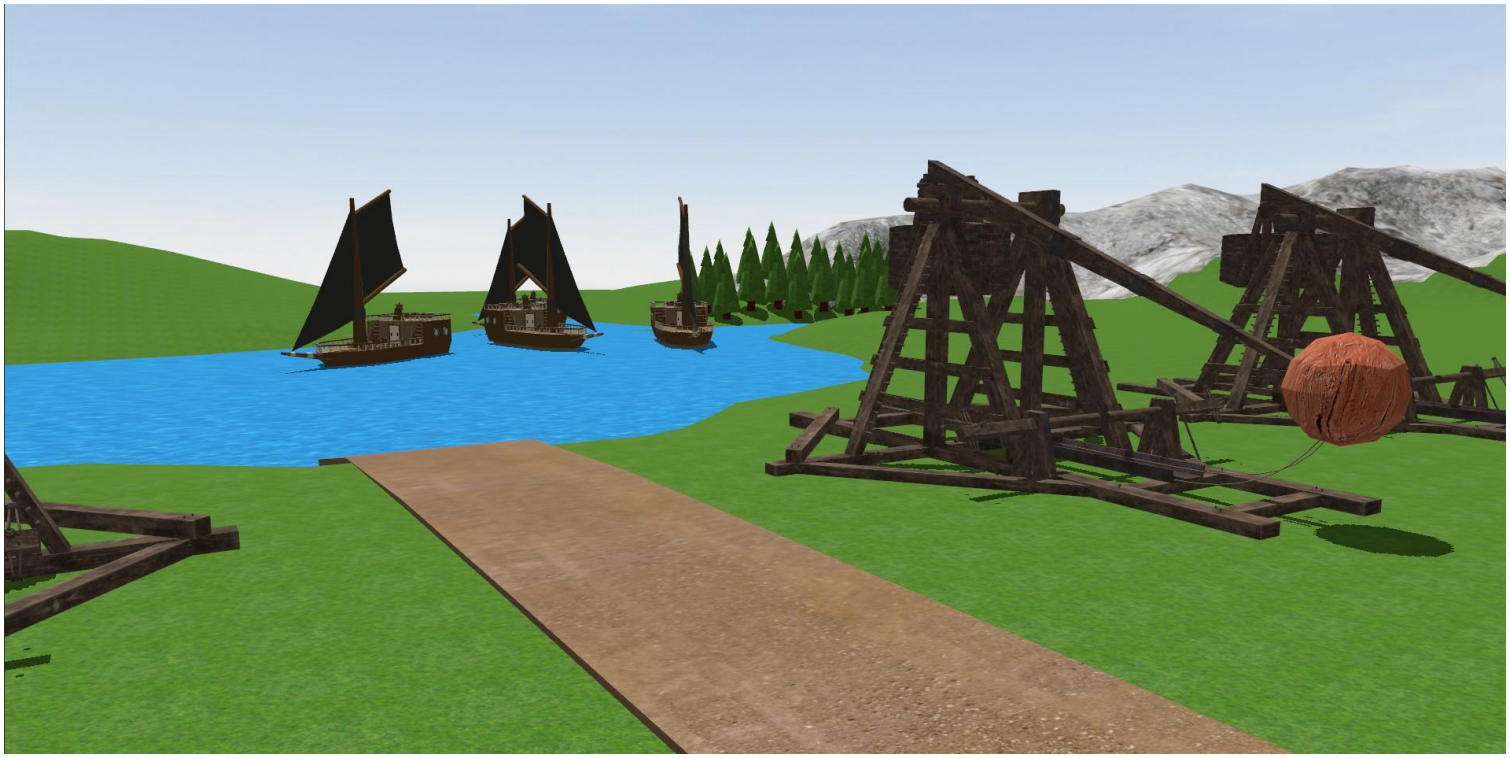
2. Scenariul

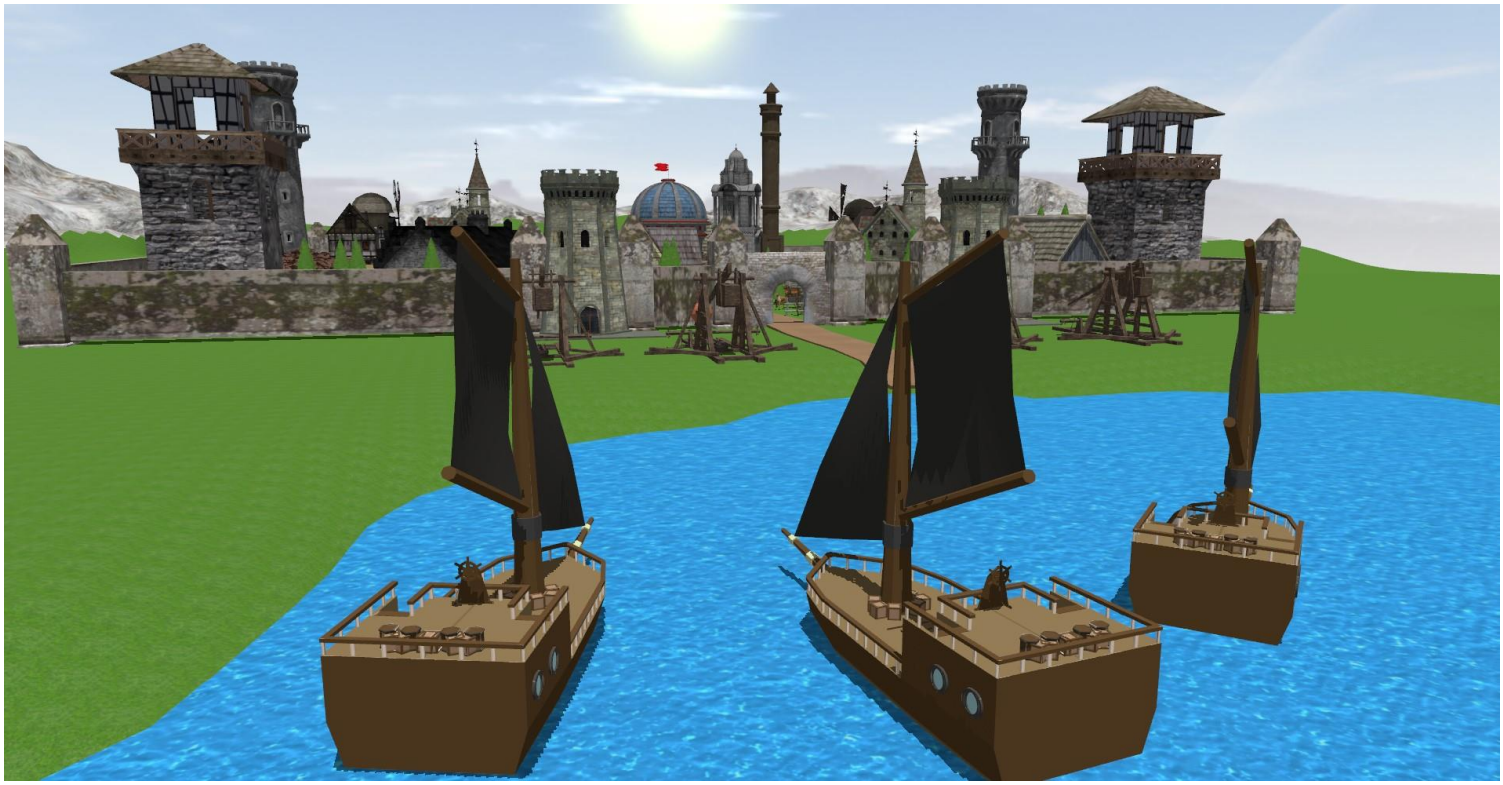
2.1. Descrierea scenei si a obiectelor

Scena reprezinta un castel medieval care incearca sa supravietuiasca unui atac al piratilor. Castelul este aparat de ziduri si turnuri din care oamenii pot anticipa mai bine pericolul. La intrare, se pot observa 4 catapulte cu care soldatii pot opri inaintarea navelor pe apa. Satul din interiorul castelului are o forma de stea, cu stradute care duc spre fiecare loc important. In mijlocul acestuia se poate observa o piata, cu tarabe cu fructe si legume, de unde oamenii isi pot cumpara cele necesare. Prin sat se gasesc numeroase fantani pentru a asigura sursa vitala de apa. Animalele sunt imprastiate prin scena, in special caii, care sunt absolut necesari oamenilor in muncile lor de zi cu zi. Se gasesc cateva case de fierar, aceasta fiind o meserie populara in vremea aceea. Se mai pot vedea caini de paza de diferite rase si de asemenea o multitudine de case care sa adaposteasca oamenii.

Mai jos se poate observa o imagine din piata din centru cu o casa de fierar

Lacul, 2 catapulte si corabiile de pirati care ataca :

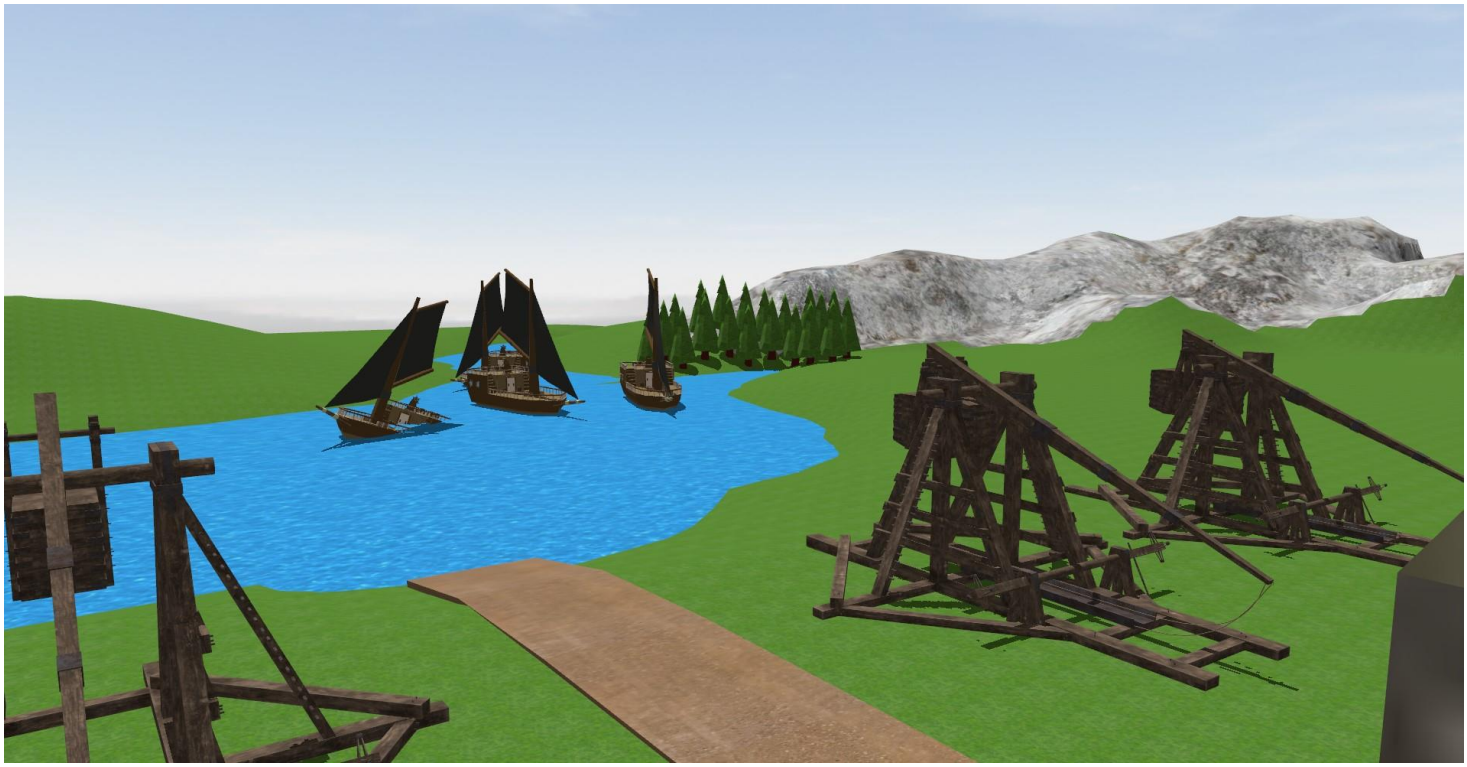




Mai jos se poate observa efectul de ceata :

Mai jos apare efectul de ceata si ploaie pe timp de zi :

Imaginea de mai jos este facuta in timp ce o corabie de pe lac se scufunda.



2.2. Functionalitati

Utilizatorul se poate misca oriunde in scena cu ajutorul tastaturii si a mouse-ului. Fereastra de vizionare se poate redimensiona si astfel imaginea sa fie mai mare sau mai mica, in functie de preferinte. In acelasi timp, scena are mai multe moduri de vizualizare : solid, wireframe sau ‘pointframe’ (adica se vad doar punctele/varfurile din care sunt formate obiectele). Un alt mod care exista este cel de vizualizare a hartii de adancime din perspectiva luminii (a soarelui), apasand tasta ‘M’

Modurile de timp :

- Utilizatorul poate alege sa seteze timpul la ‘zi’ sau la ‘noapte’. La ‘zi’, cerul va fi senin si va fi un soare pe cer care va lumina intreaga scena, iar la ‘noapte’, cerul va fi negru si va fi pe cer o luna. Lumina pe timp de ‘noapte’ va veni de la felinarele amplasate in interiorul castelului. In rest, este intuneric peste tot.

Lumina :

- In ceea ce priveste sursele de lumina, am implementat un soare, care se si poate observa ca un patrat alb pe cer, care poate fi controlat si setat de utilizator ca fiind o lumina directionala sau punctiforma. In acelasi timp, tot utilizatorul poate roti soarele in jurul castelului pentru a evidentia lumina si formarea umbrelor. Soarele apare cand este setat timpul pe modul ‘zi’. Cand este ‘noapte’, soarele ramane pe cer sub forma de luna.
- O alta sursa de lumina se poate observa doar in modul ‘noapte’. Sunt cele 8 surse punctiforme de lumina care sunt imprastiate sub forma de felinare prin interiorul castelului. In acest mod, cerul se face negru si tot ce este in exteriorul castelului este in intuneric. Aceste surse punctiforme de lumina nu produc umbra

Ceata :

- Utilizatorul poate activa sau dezactiva ceata prin tasta ‘F’. Ceata este destul de densa si se va observa imediat efectul ei

Ploaia :

- Utilizatorul poate activa sau dezactiva ploaia de pe tasta ‘R’. Se vor genera random 1000 de picaturi de ploaie care vor cadea de la inaltimi random. Acest efect formeaza un pic de lag in toata scena, lucru care se va observa usor.

Animatiile :

- Exista doua mori de vant in castel. Cea din stanga intrarii se invarte in permanenta, iar cea din dreapta poate fi controlata de utilizator. User-ul poate creste/scade intensitatea cu care elicea se invarte. Animatia acestei elice este o rotire stanga un anumit timp, dupa care dreapta acelasi timp, dupa care repeta la nesfarsit acest lucru cu viteza setata de utilizator.

Coliziunea dintre obiecte :

- Simularea coliziunii este intre o ghiulea aruncata de o catapulta si o corabie de pirati. Dupa ce este lovita, corabia se scufunda si ramane scufundata, in timp ce oamenii castelului incarca o noua ghiulea in catapulta.

3. Detalii de implementare

3.1. Functii si algoritmi

3.1.1. Solutii posibile

processMovement() - functie care se ocupa de setarirea parametrilor corespunzatori fiecarei tasta apasata

random_between(min,max) – functie care imi intoarce un numar random intre min si max.

initializa_rain() – functie care se ocupa de initializarea vectorului care tine coordonatele initiale ale picaturilor de ploaie

update_rain_coordinates() – functie care face update la fiecare randare la coordonatele picaturilor de ploaie daca este activat modul acesta. Practic se scade coordonata Y sau X a picaturii

initUniforms() – aceasta este folosita o singura data la inceputul programului pentru initializarea uniformelor din shadere

initShaders() – incarca shader-ele folosite

initObjects() – incarca obiectele folosite

windowResizeCallback() – se foloseste pentru redimensionarea ferestrei

renderScene() – acesta functie este apelata de fiecare data in main, intr-un loop, pana cand este oprita aplicatia. Functia prima data rasterizeaza scena in harta de adancime, acest lucru fiind necesar pentru calcularea umbrelor. Apoi, in functie de modul de afisare (a hartii de adancime sau a scenei cu umbre), se rasterizeaza pe ecran ceva. Daca se rasterizeaza harta de adancime a umbrelor, se foloseste 'screenQuadShader', altfel se foloseste shader-ul principal construit pentru obiectele noastre. In a doua situatie, se trimit mai intai uniformele specifice catre shadere (in afara de matricea de model), in special matricea view si alte valori care sunt afectate direct sau indirect de matricea view. Dupa ceasta etapa, se deseneaza obiectele cu functia '**drawObjects'.** In '**drawObjects'** se deseneaza mai intai scena principala, aplicandu-i transformarile specifice de model, dupa care se deseneaza obiectele individuale cu functiile descrise mai jos, unde construim o alta transformare de model, plecand de la modelul initial format pentru scena principala/de baza :

drawEliceDreapta() - deseneaza elicea din dreapta. Matricea de model are in plus o translatare in centru, dupa care rotire dupa axa X dupa care translatare inapoi la locul ei

drawEliceStanga() - deseneaza elicea din stanga. Tehnica similara cu cea descrisa la functia de mai sus

drawRainDrop() - deseneaza picaturile de ploaie. Parcurge cu un 'for' toate coordonatele picaturilor si deseneaza in fiecare coordonata cate o picatura

drawProiectil() - deseneaza proiectilul. Ghiuleaua se va misca dupa o ecuatie Bezier, din punctul in care se afla, pana intr-un punct care se afla pe o corabie. Animatia va arata ca si cum cineva ar fi lansat ghiuleaua. Pe langa modelul scenei principale, aici se mai adauga o translatie in centru de coordonate si apoi o translatie in noul punct ale carui coordonate sunt date de ecuatia Bezier. In momentul in care ghiuleaua a ajuns la destinatie, se activeaza miscarea/rotatia corabiei care incepe sa se scufunde in apa.

drawBarcaScufundata() - deseneaza barca care se scufunda. In momentul in care ghiuleaua atinge barca, aceasta isi incepe o rotatie pana cand ajunge de tot sub apa, dupa care dispare de tot sim ai poate fi desenata la loc in punctul initial doar daca utilizatorul apasa tasta ‘G’. Tehnica aici este similara cu cea utilizata pentru rotirea elicelor morilor de vant. Se translateaza in origine, se roteste, dupa care se translateaza inapoi in locatia ei originala.

Diferente intre luminile punctiforme si directionale :

- Principala diferenta este ca in cazul luminii directionale, directia este este mereu aceeasi, deoarece se considera ca este la infinit, in timp ce in cazul luminii punctiforme, directia luminii se schimba in permanenta (deoarece pozitia ei se poate schimba mereu), iar distanta fata de obiecte conteaza si influenteaza intensitatea luminii. Distanta se poate observa in cod prin calcularea valorii de atenuare in fragment shader.

Efectul de lumina punctiforma care apare doar noaptea :

- pentru acest efect, am facut 8 lumini punctiforme, toate plasate pe harta sub forma unor felinare/focuri de tabara pentru a da impresia unor strazi luminate. Tehnica folosita a fost transmiterea coordonatelor fiecarui punct de lumina din aplicatie catre fragment shader, unde pentru fiecare punct se calculeaza o valoare pentru ambientala, difuza si speculara fiecarei lumini, dupa care se aduna toate ambientalele, difuzele si specularele intre ele. Partea dificila a fost incercarea de a face luminile sa nu mai urmeasca camera in momentul in care se misca din taste. Deja era implementata partea in care luminile stau pe loc daca se roteste camera, deoarece se transmitea la shader doar locatia luminii inmultita cu o matrice de 3x3 (cea din coltul stanga sus), aceasta continand doar rotatiile, nu si translatiile. Ideea a fost deci sa transmitem shader-ului nu doar punctul inmultit cu o matrice de 3x3 cu rotatiile, ci cu una de 4x4 care sa contina atat rotatiile cat si translatiile. Acest lucru l-am implementat in aplicatie si apoi am trimis shader-ului coordonatele corecte si fata de miscarea din taste.

Efectul de umbra:

- acest efect a fost cel mai complex de realizat. Ideea consta in rasterizarea scenei de 2 ori, din doua perspective diferite. Prima trecere prin acest algoritm, adica prima rasterizare, consta din rasterizarea scenei din perspectiva luminii, generand astfel o harta de adancime intr-un frameBuffer la care am atasat o textura de adancime. In timpul acestei rasterizari, trebuie sa aducem toate obiectele relativ la lumina de la care vrem sa avem umbra,

deci avem nevoie de o matrice de transformare diferita de ‘viewMatrix’. Pentru acest lucru ne ajuta functia ‘computeLightSpaceTrMatrix’, care transmite shader-ului o matrice diferita de ‘view’ si ‘projection’. Aici proiectia se va face ortografica pentru a evita deformarile unei proiectii perspectiva. Odata ce avem harta de adancime (care contine niste valori ce ne indica adancimea celui mai apropiat fragment de lumina), putem trece la a doua rasterizare, si anume la cea din perspectiva camerei, tinand insa cont de harta de adancime abia creata. In cadrul acestei rasterizari, ideea este ca fiecare fragment sa fie adus din nou la referinta luminii, sa se compare adancimea lui cu valoarea de adancime de la aceleasi coordonate din harta de adancime si daca are adancimea mai mare, inseamna ca fragmentul este in umbra. Daca fragmentul este in umbra, atunci vom ignora componenta difuza si speculara a acestuia.

Efectul de coliziune a ghiulelei si a barcii:

- Miscarea ghiulelei a fost realizata cu ajutorul unei ecuatii Bezier, folosind 3 puncte: P0 – punctul de plecare ; P1 – punctul intermediar; P2 – punctul final.

```
glm::vec3 P0(0.418016f, 0.113617f, 2.503f);//punct de plecare
glm::vec3 P1(0.503596f, 1.82396f, 4.16872f);//punct intermediar
glm::vec3 P2(0.566333f, -0.027702f, 5.3729f);//punct final
glm::vec3 translate_vector = (P2 * t + P1 * (1 - t)) * t + (P1 * t + P0 * (1 - t)) * (1 - t);
```

Mai sus se poate observa formula dupa care se misca ghiuleaua. Variabila ‘t’ creste de la 0.0f la 1.0f cu un factor de 0.005f. In momentul in care trece de 1.0f, aceasta animatie se opreste si se porneste animatia barcii. Aceasta consta in rotatia (incadrata intre 2 translatii care aduce obiectul in origine si dupa il duce inapoi in punctul lui original) pe axa Z, pana cand atinge un anumit numar, semn care ne indica ca, corabia a ajuns sub apa. Corabia poate fi redesenata cu era initial daca se apasa tasta ‘G’.

Efectul de rotatie a elicei din dreapta :

Aceasta elice se roteste intr-o directie pana cand unghiul de rotire ajunge la o anumita valoare, dupa care se roteste in directia opusa tot pana la aceeasi valoare in modul.

Efectul de ceata :
Efectul de ceata a fost realizat in fragment shader. Acesta poate fi activat de la tasta ‘F’. Am folosit urmatoarea formula petru a calcula factorul de ceata, care apoi va contribui la o interpolare intre culoarea cetii si culoarea fragmentului. Rezultatul reprezinta noua culoare a fragmentului.

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

3.1.2. Motivarea abordarii alese

Am ales aceasta abordare deoarece majoritatea functiilor erau deja implementate la laborator, deci am ales sa incep proiectul de la un proiect in care erau facute mai multe laboratoare. Din punctul acesta, am construit aplicatia si am adaugat functionalitatile lipsa din arhiva cu proiectul incarcata pe moodle.

3.2. Modelul grafic

Obiectele si texturile au fost in amre parte descarcate de pe interner, importate in scena si plasate in blender. Majoritatea aveau textura deja prezenta in fisiere lor, iar celor care nu, precum brazii, le-am construit-o de la zero.

Alte obiecte facute/construite sunt:

- picaturile de ploaie: au fost construite/modelate dintr-un patrat, alungit cat sa semene cat mai mult cu o picatura. Avantajul este ca avand doar 8 varfuri, nu influenteaza atat de mult rasterizarea scenei (deoarece trebuie sa se desene foarte multe astfel de picaturi)

3.3. Structuri de date

In acest proiect nu am folosit alte structure de date aditionale pe langa cele déjà aflate in arhiva proiectului sau a laboratoarelor. S-au folosit structuri din bibliotecile glm, glfw si glew.

3.4. Ierarhia de clase

- **Camera –** contine implementarea pentru miscare si rotatie camerei, plus niste getter-e si setter-e care ajuta la modul ‘presenter’
- **Main –** contine partea principala de cod
- **Mesh**
- **Model3D –** se foloseste la citirea obiectelor si incarcarea texturilor
- **Shader –** se ocupa cu incarcarea shader-elor
- **Skybox –** contine implementare skybox-ului
- **Stb_image**
- **Tiny_obj_loader**
- **Window –** se ocupa cu actiunile legate de fereastra

4. Prezentarea interfetei grafice utilizator/ manual de utilizare

- miscarea camerei:
    - **W -** miscarea camerei inainte
    - **A -** miscarea camerei spre stanga
    - **S –** miscarea camerei inapoi
    - **D –** miscarea camerei in dreapta
- controlul timpului:
    - **B –** setez timpil la ‘zi’
    - **N –** setez timpul la ‘noapte’

- controlul luminilor:
    - **Y –** setez soarele ca lumina directionala
    - **U –** setez soarele ca lumina punctiforma
    - **J –** scade unghiul de rotire al soarelui (soarele se roteste in jurul castelului)
    - **K –** creste unghiul de rotire al soarelui (soarele se roteste in jurul castelului)
- controlul efectelor:
    - **R –** activeaza/dezactiveaza ploaia
    - **F** – activeaza/dezactiveaza ceata
    - **P –** cresc viteza de rotatie a elicei din dreapta
    - **O –** scad viteza de rotatie a elicei din stanga
    - **T –** activez miscarea proiectilului, care singur dupa activeaza miscarea corabiei
    - **G –** activez desenarea barcii in cazul in care s-a scufundat si vreau sa testez din nou coliziunea obiectelor
- controlul rotatiei scenei:
    - **Q –** scade unghiul de rotire a intregii scene (intreaga scena cu exceptia soarelui/lunii)
    - **E –** scade unghiul de rotire a intregii scene (intreaga scena cu exceptia soarelui/lunii)
- controlul modurilor:
    - **M –** activez/dezactivez modul de afisare al shadow map-ului
    - **L –** activez/dezactivez modul ‘presenter’
    - **V** – activez miscarea completa a mouse-ului in scena
    - **Z –** activez modul normal – cel cu textura obiectelor
    - **X –** activez modul wire-frame
    - **C –** activez modul de afisare a varfurilor din care sunt construite obiectele

5. Concluzii si dezvoltari ulterioare

Ca o concluzie generala, pot sa spun ca mi-a facut mare placere sa lucrez la acest proiect si consider ca am inteles mult mai bine o multime de notiune de matematica invatate in trecut la alte materii. A fost interesant sa modelez obiecte in blender si sa alcatuiesc o scena reala cu acestea.

6. Referinte

Linkuri pentru obiecte :

- https://www.turbosquid.com/
- https://free3d.com/

In rest, singurele referinte au fost lucrarile de laborator si cursurile