Scenă 3D - Candy Land

Vlad Ursache

Cuprins

Rezumat

Tema aleasă este Candy Land, un tărâm vrăjit populat de bomboane și unicorni pofticioși de bunătățuri zaharisite și SÂNGE. Atmosfera dulce este repede înlocuită cu una apăsătoare.

1 Scenariul

1.1 Descrierea scenei și a obiectelor

Terenul este făcut din napolitană, iar peste acesta sunt plasate obiecte din cele mai bizare, care se mai încadrează totuși în temă. Printre aceste obiecte se numără unicornul de pluș din centrul scenei, acadeaua rotitoare și șarpele devorator de gogoși glazurate. Râul este de ciocolată și pe el plutește un unicorn gonflabil care vânează cornetele de înghețată de la o distanță sigură. Peste ciocolată putem vedea un pod și altă bomboană rotitoare care unesc insulițele ce compun scena. Luna însângerată luminează calea.
Dar noaptea se așterne și o ceață neagră, misterioasă acaparează scena. Adevăratele intenții ale tuturor devin clare.
Obiectiv principal: Scapă cu viață! Puncte bonus dacă explorezi tenebrele Candy Land! Mai multe puncte bonus dacă te-ai și distrat!

1.2 Funcționalități

Exploratorul se poate folosi de tastele W, A, S, D pentru a se mișca prin scenă cu ajutorul camerei first person. Tastele Z, X și C schimbă scena în modul de vizualizare al muchiilor, vârfurilor și modul normal. Tasta V activează și dezactivează ceața și lumina roșie. Sursa de lumină poziționlă (luna însângerată) se rotește singură, alături de acadea și bomboana pod. Tasta M afișează scena din perspectiva luminii poziționale.
De asemenea, un fragment audio este redat la execuție, pentru aproximativ 20 de secunde, cu scopul de a tensiona atmosfera.

2 Detalii de implementare

2.1 Funcții și algoritmi

Funcțiile keyboardCallback(), mouseCallback() si processMovement() se ocupă de interacțiunea utilizatorului cu aplicația prin intermediul tastaturii si a mouse-ului. Acestea au fost implementate în laboratorul care folosește first person camera. O verificare suplimentară a fost adăugată pentru a elimina Gimbal lock, o stare în care rotația camerei pierde un grad de libertate.

```
if (pitch > 89.0f)
    pitch = 89.0f;
if (pitch < -89.0f)
    pitch = -89.0f;
```

Funcția initOpenGLWindow() este responsabilă pentru crearea ferestrei în rezoluția dorită, dar și de pornirea fragmentului audio pentru cutscene-ul de început. Un API disponibil doar pe platforma Windows a fost utilizat.
setWindowCallbacks() setează callback-urile. Am schimbat input mode-ul la mouse pentru a nu apărea probleme la mișcarea camerei. Acum poate executa rotații complete, iar cursor-ul nu mai este vizibil. initOpenGLState() este o altă funcție de configurare a parametrilor scenei. Din cauza unor obiecte care au normalele inversate, am dezactivat cull face-ul pentru a le rasteriza corect.
Funcția initSkyBox() încarcă cele 6 fețe ale skybox-ului și le încarcă în aplicație. Acompaniate de parametrii ceruți de shader, skybox-ul așteaptă să fie desenat. initModels() încarcă în mod similar obiectele care vor apărea pe scenă. initShaders() pregătește shaderele pentru a fi ulterior folosite. initUniforms inițializează variabilele de tip uniform spre a fi ulterior folosite de shadere.
initFBO() creează un FrameBuffer Object, o textură de adâncime. Această textură este ulterior atașată FBO-ului. Pentru a genera umbrele, este nevoie și de matricea de transformare în spațiul luminii (pentru a vizualiza scena din perspectiva sursei de iluminare poziționlă).
presentScene() se ocupă cu mișcarea automată a camerei la cutscene-ul de început. Acesta se folosește de o variabilă care simulează trecerea timpului pentru tranziții smooth.
renderModels() rasterizează obiectele din scenă cu ajutorul shaderelor basic, care implementează modelul de iluminare Phong. Dacă este activat modul horror, sunt trimise la shader-ul basic

si de skybox variabilele uniform responsabile pentru schimbarea culorii luminii direcționale și a ceții. Dacă rasterizăm depth map-ul, atunci nu trimitem matricea normală. Obiectele care suferă transformări sunt animate în această funcție. Le sunt aplicate matrici de transformare matricii model.
În renderScene() este logica de generare a shadow map-ului (similar cu ZBuffer) și rasterizarea scenei întregi, cu tot cu umbre. Aceasta se schimbă în funcție de poziția lunii însângerate, care se rotește deasupra noastră.
cleanup() conține codul de curățare a datelor pe care le-am inițializat.

2.1.1 Soluții posibile

Implementarea este una cât se poate de apropiată de laborator. Cu excepția animării unor obiecte și a shaderelor basic si skybox, nu sunt abateri majore de la codul de la laborator.
Obiectele care se rotesc sunt centrate inițial în origine. Asupra lor se efectuează rotație in jurul axei Y (în OpenGL) și translație la locația dorită.
Shader-ul basic implementează modelul de iluminare Phong, ceață și umbre. Implementarea acestuia este una convențională. Shader-ul skybox schimbă culoarea acestuia în funcție de valoarea variabilei uniform pentru ceață. Astfel, ceața afectează atât obiectele relativ apropiate, dar și fundalul foarte îndepărtat pentru un efect consistent.

2.1.2 Motivarea abordării alese

Toate implementările se bazează pe un principiu simplu: orice poate fi implementat. Dat fiind timpul limitat, aleg funcționalitățile cele mai interesante și le implementez corect, în cea mai simplă manieră posibilă.

2.2 Modelul grafic

Următoarele printscreen-uri sunt direct din aplicația OpenGL. Nu au suferit modificări în post producție.

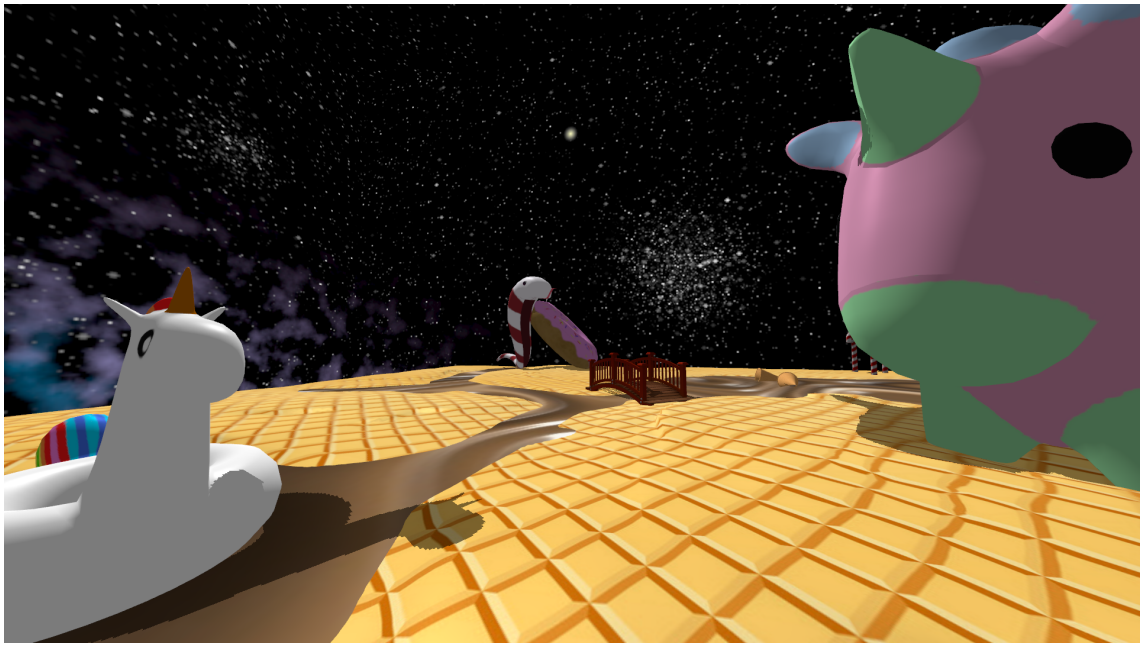

Figura 1: Scena în modul "fill"

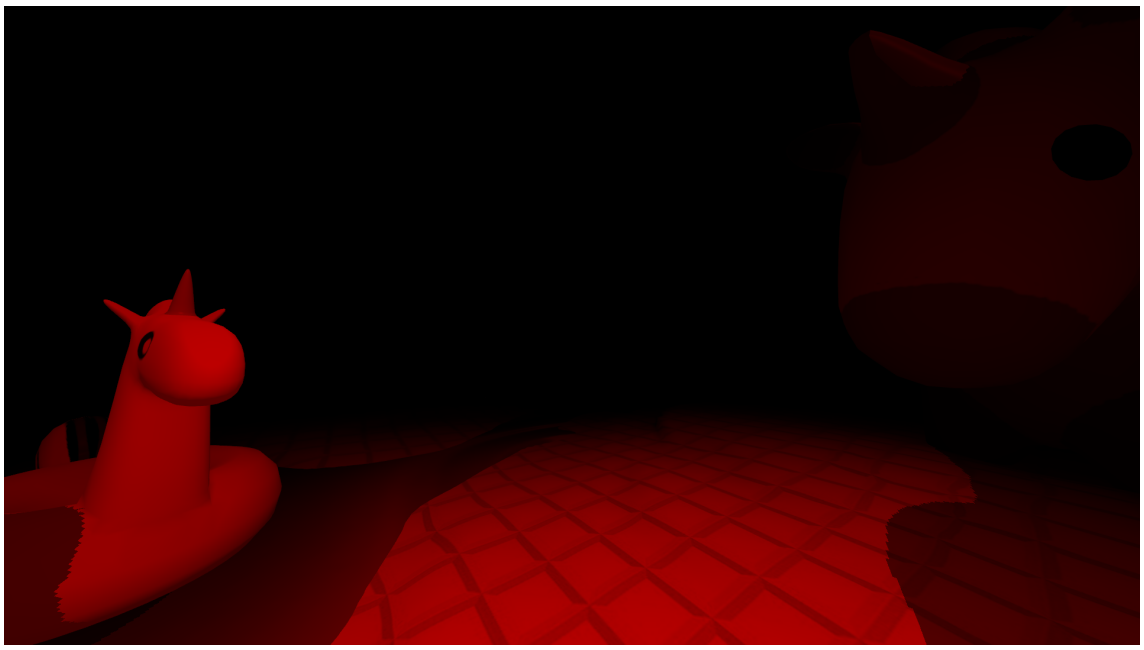

Figura 2: Scena în modul ”horror”

Umbrele sunt poziționate diferit în cele 2 exemple.

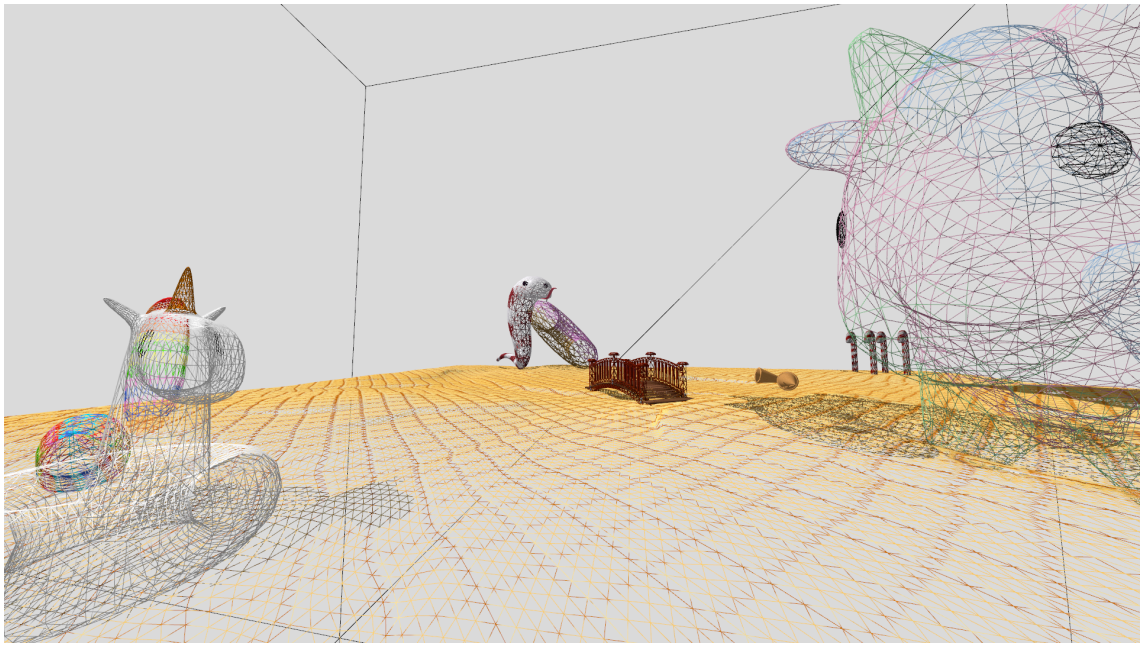

Figura 3: Scena în modul ”line”

Figura 4: Scena în modul ”point”

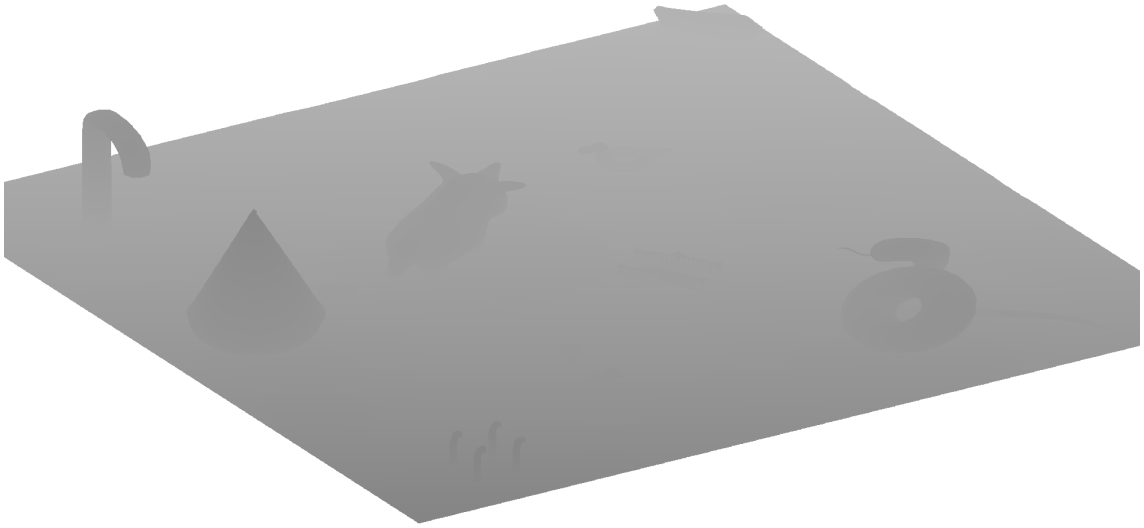

Figura 5: Scena în depth map (din perspectiva luminii)

2.3 Structuri de date

Au fost folosite clasicele structuri de date specifice OpenGL: vec3, vec4, mat3, mat4 (specifice glm), dar și obiecte instanțiate din clasele noastre: Camera, Model3D, Shader, SkyBox. Am folosit și tipurile de dată specifice glm: GLuint și GLfloat, care au compatibilitate superioară cu OpenGL.

2.4 Ierarhia de clase

Lista fișierelor din codul sursă este următoarea:

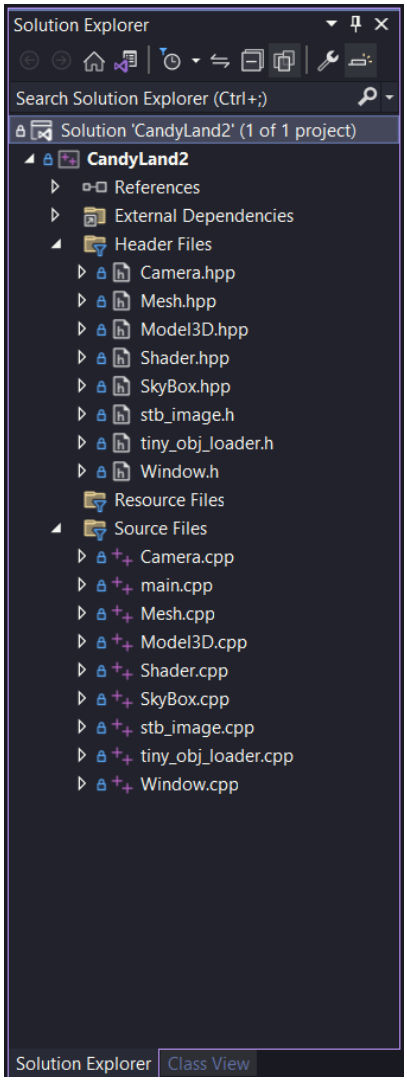

Figura 6: Listă de clase

3 Prezentarea interfeței grafice utilizator / manual de utilizare

A se vedea secțiunea 2.2.

4 Concluzii și dezvoltări ulterioare

În concluzie, scena 3D este operațională:

- Scena este navigabilă cu ajutorul tastaturii și a mouse-ului, dar și cu ajutorul animației de prezentare (sub formă de cutscene, la început de execuție).
- Am două surse de lumină, una direcționala și una pozițională (luna însângerată).
- Vizualizarea exclusivă a muchiilor și a vârfurilor funcționează. Putem de asemenea vizualiza depth map-ul.
- Texturile obiectelor funcționează în modul în care le-am gândit.

- Umbrele sunt generate și sunt dinamice.
- Există obiecte animate în scenă.
- Fotorealismul și preferințele personale sunt subiective.

Printre dezvoltările ulterioare se numără:

- Modificarea obiectelor din scenă pentru mărirea calității scenei.
- Umplerea scenei cu detalii care să umple spațiul liber.
- Reconturarea reliefului scenei.
- Adăugarea transparenței și efectului de valuri asupra râului de ciocolată.
- Adăugarea logicii de coliziune.
- Adăugarea unui personaj și a unei camere third person care sa-l surprindă.
- Adăugarea unor surse de lumină portocalii fluctuante, cum ar fi o torță sau un foc de tabără.

A Referințe

Toate resursele utilizate sunt cele din laboratoarele de Prelucrare Grafică ale Departamentului de Grafică al UTCN. Asset-urile utilizate sunt împrumutate de pe diverse site-uri de modele 3D de pe Internet, preponderent https://sketchfab.com/.