



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

3D Project

Proiectare Grafica

Autori: Zubascu Maria

Grupa: 30234

FACULTATEA DE AUTOMATICA
SI CALCULATOARE

15 Ianuarie 2024

Cuprins

1 OpenGL	2
1.1 Introducere	2
2 Blender	2
2.1 Introducere	2
3 Specificatii	2
4 Descrierea scenei si a obiectelor	3
5 Functionalitate	5
6 Implementare	6
6.1 Functii	6
6.1.1 Solutii posibile	6
6.1.2 Motivarea abordarii alese	7
7 Modelul Grafic	8
8 Structuri de date	8
9 Ierarhia de clase	8
10 Ghid de utilizare	8
11 Concluzii si dezvoltari ulterioare	8
12 Bibliografie	8

1 OpenGL

1.1 Introducere



OpenGL (Open Graphics Library) este o specificație a unui standard care definește un API (Application Programming Interface) multiplatformă, foarte utilizat pentru programarea componentelor grafice 2D și 3D ale programelor de calculator. Interfața constă în peste 250 de apeluri diferite care folosesc la a desena pe ecranul calculatorului scene 3D complexe din primitive (din primitives, elemente simple). OpenGL a fost inițial dezvoltat de compania Silicon Graphics, Inc. (SGI) în 1992 și este foarte utilizat în grafică asistată de calculator, realitate virtuală, vizualizare științifică, simulări de zboruri sau jocuri pe calculator. Acest ultim domeniu este în strânsă competiție cu tehnologia DirectX de la Microsoft (compară OpenGL cu Direct3D). Proiectul OpenGL este condus de compania Khronos Group, un consorțiu tehnologic non-profit.

2 Blender

2.1 Introducere



Blender este un set de instrumente software gratuite și open-source pentru grafică 3D, utilizat pentru crearea de filme de animație, efecte vizuale, artă, modele imprimate 3D, grafică de mișcare, aplicații 3D interactive, realitate virtuală și, anterior, jocuri video. Printre caracteristicile Blender se numără modelarea 3D, cartografierea UV, texturarea, desenul digital, editarea grafică raster, rigging și skinning, simularea fluidelor și a fumului, simularea particulelor, simularea corpului moale, sculptarea, animația, mișcarea meciurilor, randarea, grafica de mișcare, editarea video și compoziția.

3 Specificații

Acest proiect are ca scop crearea unei reprezentări realiste a obiectelor 3D folosind biblioteca OpenGL, un API utilizat pe scară largă și transplatformat pentru programarea graficii 2D și 3D în programele de calculator.

Am ales să creez o scenă care reprezintă un Camping în desert, prin care traversează un râu, în jur se află camile, palmieri, piramide și un OZN.

4 Descrierea scenei si a obiectelor

Scena este in mijlocul unui desert de-a lungul caruia traverseaza un rau. Pe malul raului se afla un spatiu verde cu o ”padure” de palmieri, imediat langa acest spatiu verde se afla nisip pe care sunt cateva corturi asezate in jurul unui foc de tabara. Acest loc de campat se afla chiar langa un lant de piramide pe langa care trec 4 camile in sir indian. La malul lacului se afla, de asemenea, doua barci de lemn. Deasupra acestui loc de camping este un OZN care se poate deplasa (sus, jos, stanga, dreapta). Intreg cadrul este intr-un fundal de apus.







5 Functionalitate

Utilizatorul poate vedea o previzualizare completă a scenei înainte de a începe să se deplaseze liber pe hartă. El poate avea o vizualizare a întregii scene prin apăsarea tastelor de la tastatură

și prin utilizarea mouse-ului pentru a direcționa camera în poziția dorită. Obiectul OZN plasat în aer poate fi translatat (sus, jos, dreapta, stanga), iar scena poate fi vizualizată în modurile solid, wireframe, poligonal, apasând tastele corespunzătoare.

6 Implementare

6.1 Functii

Cele mai importante functii ale acestui proiect sunt:

- initOpenGLState() : pentru a activa unele funcții specifice OpenGL
- initModels() : unde toate modelele sunt încărcate din fișierele corespunzătoare
- initShaders(): în care sunt încărcate toate shaders-urile
- setWindowCallbacks() : activează callback-urile pentru fereastră, tastatură și mouse
- processMovement() : procesează acțiunile de la taste
- renderScene() : pentru redarea obiectelor cu shaderele corespunzătoare (în interiorul acestei funcții avem apele la funcția renderObjects(gps::Shader shader, bool depthPass), care redă obiectele noastre în funcție de parametri

6.1.1 Solutii posibile

In cadrul acestei functii am adaugat copaci in scena, acestia fiind pe doua plane, la departare dispar parti din ei, asa ca am folosit functiile glDisable, glEnable pentru a rezolva aceasta problema

```

1 void renderCopaci(gps::Shader shader) {
2     // select active shader program
3     shader.useShaderProgram();
4
5     //send teapot model matrix data to shader
6     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
7
8     //send teapot normal matrix data to shader
9     glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));
10
11    glDisable(GL_CULL_FACE);
12    // draw teapot
13    copaci.Draw(shader);
14    glEnable(GL_CULL_FACE);
15 }
```

In cadrul acestei functii am introdus obiectul OZN in scena, la apasarea butoanelor Y, G, H, J acesta va fi translatat in stanga, dreapta, sus, jos cu ajutorul functiei glm::translate()

```

1 void renderOzn(gps::Shader shader) {
2
3     // update model matrix for teapot
4     oznmodel = glm::translate(model, oznPosition);
5     // update normal matrix for teapot
6     oznMatrix = glm::mat3(glm::inverseTranspose(view * oznmodel));
```

```

7
8     if (pressedKeys[GLFW_KEY_Y]) {
9         oznPosition.y += oznSpeed;
10        // update model matrix for teapot
11        oznmodel = glm::translate(model, oznPosition);
12        // update normal matrix for teapot
13        oznMatrix = glm::mat3(glm::inverseTranspose(view * oznmodel));
14    }
15    if (pressedKeys[GLFW_KEY_H]) {
16        oznPosition.y -= oznSpeed;
17        // update model matrix for teapot
18        oznmodel = glm::translate(model, oznPosition);
19        // update normal matrix for teapot
20        oznMatrix = glm::mat3(glm::inverseTranspose(view * oznmodel));
21    }
22    if (pressedKeys[GLFW_KEY_G]) {
23        oznPosition.x -= oznSpeed;
24        // update model matrix for teapot
25        oznmodel = glm::translate(model, oznPosition);
26        // update normal matrix for teapot
27        oznMatrix = glm::mat3(glm::inverseTranspose(view * oznmodel));
28    }
29    if (pressedKeys[GLFW_KEY_J]) {
30        oznPosition.x += oznSpeed;
31        // update model matrix for teapot
32        oznmodel = glm::translate(model, oznPosition);
33        // update normal matrix for teapot
34        oznMatrix = glm::mat3(glm::inverseTranspose(view * oznmodel));
35    }
36
37    // select active shader program
38    shader.useShaderProgram();
39
40    // send ozn model matrix data to shader
41    glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(oznmodel));
42
43    // send ozn normal matrix data to shader
44    glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(oznMatrix));
45
46    // draw scene
47    ozn.Draw(shader);
48 }

```

6.1.2 Motivarea abordarii alese

Pe parcursul acestui laborator ne-au fost prezentate diferite metode de a crea o scena 3D. Am ales sa proiectez scena si obiectele in Blender deoarece am avut un suport didactic foarte bun, am implementat functionalitatile cu usurinta deoarece am primit explicatii clare in cadrul

laboratorului. De asemenea, indrumatorul de laborator si sablonul pe care l-am primit pentru proiect au fost de foarte mare ajutor.

7 Modelul Grafic

Obiectele și texturile au fost descărcate de pe Google. Majoritatea obiectelor au fost importate în Blender pentru a fi editate și plasate în scena finală. Scena a fost, de asemenea, realizată în Blender, după care a fost adăugată în OpenGL.

8 Structuri de date

Structurile de date utilizate în acest proiect sunt structurile de date comune utilizate în C/C++ și structurile de date din biblioteca glm (cum ar fi vectori, matrici etc.).

9 Ierarhia de clase

- Camera: conține implementarea mișcării camerei (sus, jos, față, spate, stânga, dreapta)
- Mesh
- Model3D
- Shader: conține metode de creare și activare a programelor de shader

10 Ghid de utilizare

- miscarea mouse-ului - pozitionarea camerei
- W, S, A, D - miscarea camerei (față, spate, stânga, dreapta)
- R - ridică camera
- F - coboară camera
- Q, E - rotatia camerei în stanga si in dreapta
- Y,G,H,J - deplasarea obiectului OZN (sus, jos, stanga, dreapta)
- Z - modul solid
- X - modul wireframe
- C - modul poligonal
- V - modul night
- B - revenire la modul normal

11 Concluzii si dezvoltari ulterioare

In concluzie, acest proiect constă în proiectarea și implementarea unei scene 3D în limbajul de programare C/C++. Codul a fost implementat în Microsoft Visual Studio. Posibile dezvoltări ar fi adăugarea umbrelor, foc în locul focului de tabara, o lumină albastră în jurul OZN-ului, apă curgătoare, oameni. Acest proiect ar putea fi scena unui joc în care extratereștrii atacă un camping în desert.

12 Bibliografie

<https://free3d.com>

indrumatorul de laborator

<https://learnopengl.com/>