Mihoc Tudor Dan
MLR5060 - Grafică pe calculator
March 17, 2022

Laboratorul 2

Rezolvați exercițiile de mai jos. Termenul de predare este Laboratorul 3. Se poate întârzia cu o penalizare de 2 puncte până la Laboratorul 4.

Exerciții

Exercițiul 1. Desenați o sferă. Folosiți exemple date pentru a va inspira.

- folosind **fixed function pipeline** – exemplu în *cub_fp.zip*
- folosind **programmable pipeline** – exemplu în *cub_pp.zip* Incercați să folosiți

Exercițiul 2. Imprimați obiectului o mișcare de rotație în jurul unei axe proprii precum și o mișcare de revoluție în jurul unui punct pe o traiectorie circulară.
ATENȚIE! pentru a face așa ceva în *programmable pipeline* trebuie folosite variabile *uniforms*.

Apendix

Pentru a desena obiectele 3D trebuie precizate șirurile de vertexuri.

Exemplul 1: Cubul

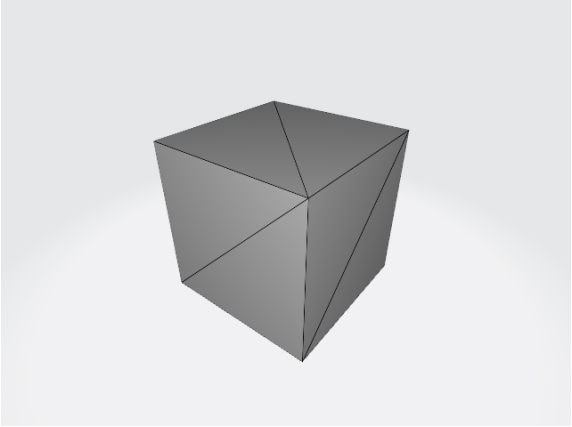

FIGURE 1. Un cub cu latura unitate; observați triunghiurile cu care formăm fețele cubului.

Pentru a desena un cub, considerăm mulțimea de vertexuri din tabela 1 și multimea de triunghiuri din tabela 2.

v_1	1.000000	1.000000	-1.000000
v_2	-1.000000	-1.000000	-1.000000
v_3	-1.000000	-1.000000	1.000000
v_4	1.000000	-1.000000	-1.000000
v_5	-1.000000	1.000000	1.000000
v_6	1.000000	1.000000	1.000000
v_7	1.000000	-1.000000	1.000000
v_8	-1.000000	1.000000	-1.000000

TABLE 1. Vertexurile necesare pentru a desena un cub (numărul lor e redus la minim datorită faptului că nu ne interesează culorile fețelor, sau normalele la suprafețe sau alte elemente – deocamdată).

f_1	7	5	0
f_2	5	2	6
f_3	4	1	2
f_4	3	2	1
f_5	0	6	3
f_6	7	3	1
f_7	7	4	5
f_8	5	4	2
f_9	4	7	1
f_{10}	3	6	2
f_{11}	0	5	6
f_{12}	7	0	3

TABLE 2. Triunghiurile ce formează cubul.

În exemplul *cub_pp.zip* cubul este construit folosind aceste date. Observați modul cum sunt transmise vertexurile către shadere.

Diferențele principale din codul unde am desenat triunghiul si acest exemplu sunt:

1) definim spațiul în care lucrăm cu ajutorul matricilor *model*, *view*, *projection*. Ne vom folosi pentru asta de biblioteca de funcții **glm**;
2) folosim pentru simplificare și un vector EBO care ne va permite să precizăm indicii vertexurilor ce formează fiecare triungi (Tabela 2);
3) transmitem valorile matricilor programului prin metode specifice (variabile de tip uniforms);
4) folosim funcția *glDrawElements* în locul la *glDrawArrays*;
5) realizăm o rotație a obiectului pentru a-l vedea mai bine.

Atenție! Viteza de rotație a cubului este optimizată pentru unul din calculatoarele mele, jucați-vă cu ea pentru o animație plăcută pe calculatorul vostru.

Exemplul 2: Calotă sferică

Să presupunem că vrem să desenăm o calotă sferică ce aparține sferei cu raza de o unitate $R = 1$ și are înalțimea $h = 0.45$.

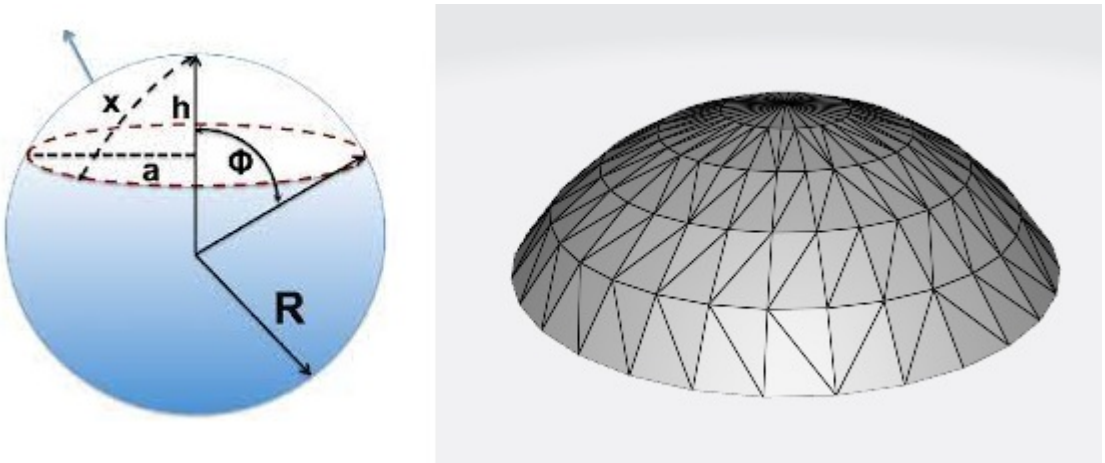

FIGURE 2. O calotă sferică cu raza sferei R, raza calotei a, înălțime h, unghiul φ și mesh-ul creat de vertexuri cu care poate fi reprezentată.

Putem genera vertexurile și triunghiurile în mai multe moduri:

1 construim modelul 3D în Blender (de exemplu) și îl importăm cumva (laboratorul următor);
2 folosind ecuațiile parametrice ale sferei;

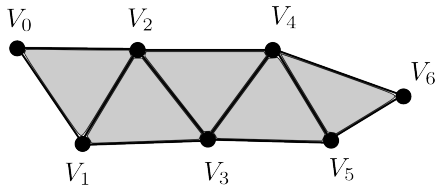

FIGURE 3. Triangle strip. Vertexurile vor fi luate în ordine v_0, v_1, ..., v_6.

3 descriem locul geometric a punctelor de pe calotă - ca mulțime de puncte egal depărtate de centrul sferei;
4 transformând un icosaedru;
- altele ...

Să considerăm o sferă cu centrul în originea sistemului de referință.
Un punct de pe aceasta sferă are coordonatele:

$$A(x, y, z) = (R * \cos\theta \sin\varphi, R * \sin\theta \sin\varphi, R * \cos\varphi)$$

unde θ și φ sunt unghiurile făcute de vectorul $\overline{OA}$ cu axele Ox și respectiv Oz.
Pentru o calotă cu raza a trebuie să calculăm limitele între care trebuie să fie unghiul φ.

Dacă $\alpha = \arcsin\frac{a}{R}$ atunci $\varphi \in [\alpha, \pi]$ și unghiul $\theta \in [0, 2\pi]$.
Vertexurile vor fi în punctele de la intersecția a n paralele și a m meridiane echidistante obținute cu unghiuri din aceste două intervale.
Triunghiurile pot fi formate cu primitiva GL_TRIANGLE_STRIP (ca în figura 3) aranjând punctele astfel încât să se parcurgă triunghiurile cuprinse între două paralele.
Pentru asta nu vom mai avea nevoie de EBO ci doar de VAO și VBO. Vom construi un vertex array care îl vom reprezenta cu ajutorul funcției *glDrawArrays* similar cu modul cum am desenat triunghiul.
Aveți în exemplul *calota.zip* o variant de a desena o sfera în acest mod.

Bibliografie

[1] `http://www.songho.ca/opengl/gl_sphere.html`

[2] `https://learnopengl.com/`