Incarcarea modelelor

Incarcarea datelor in buffer

Pentru a nu fi nevoiti sa adaugam in buffer de mai multe ori vertecsii comuni mai multor fete, ne vom folosi de un index buffer. Astfel vom avea doua tipuri de buffere:

- Cel pentru vertecsi, de tip GL_ARRAY_BUFFER
- Cel pentru indici, de tip GL_ELEMENT_ARRAY_BUFFER

In aceste doua buffere vor fi pusi vectorii obtinuti in urma parsarii fisierului nfg (vectorul de vertecsi si cel de indici).

Desenarea

Pentru a desena cu ajutorul indicilor, vom folosi *glDrawElements* in loc de *glDrawArrays*.

glDrawElements(mod_preluare_vertecsi, numar_indici, tip_date_indici, pointer_locatie_indici);

Deci in cazul nostru vom avea:
glDrawElements(GL_TRIANGLES, nr_indici, GL_UNSIGNED_SHORT, 0);

Am pus 0 pe ultimul parametru deoarece, avand datele stocate in buffer, ultimul parametru capata rol de offset si nu de locatie de unde sa ia datele. Daca nu vrem sa folosim un buffer pentru indici, punem acolo adresa vectorului de indici.

Cand facem apelul *glDrawElements*, ambele buffere (si cel de vertecsi si cel de indici) trebuie sa fie deschise.

Vedeti si:

- https://www.khronos.org/opengles/sdk/docs/man/xhtml/glDrawElements.xml
- http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-9-vbo-indexing/ (ce e scris acolo e valabil si pentru OpenGL ES)

Incarcarea texturilor

Texturile sunt imagini care se aplica pe modelul 3D. Ni le putem imagina ca pe niste abtibilduri pe care le putem redimensiona si din care putem decupa ce avem nevoie ca sa putem lipi imaginea rezultata pe o fata a modelului.

De citit despre texturi:
http://blog.digitaltutors.com/cover-bases-common-3d-texturing-terminology/
http://blog.digitaltutors.com/understanding-difference-texture-maps/

http://blog.db-in.com/all-about-opengl-es-2-x-part-2#textures

Pentru a specifica ce portiune de textura e asociata unei fete de pe suprafata modelului, se asociaza fiecarui vertex un set de **coordonate u,v** din intervalul [0,1] (pot fi si din afara acestui interval insa vor fi tratate in mod special, asa cum e explicat mai jos). Coordonatele u,v pentru fragmente sunt calculate prin interpolare. Intervalul [0,1] este mapat pe latimea si, respectiv, inaltimea imaginii-textura. Astfel, punctul de culoare specificat prin coordonatele u,v subunitare va fi egal cu **texelul** din imagine de la coordonatele (u*lungime,v*inaltime).

Pentru a genera un id de textura:
glGenTextures(nr_texturi, locatie_in_care_sa_fie_stocat_id-ul/id-urile)
Exemplu:
glGenTextures(1, &id_textura);

Ca si in cazul bufferelor, textura trebuie "deschisa" pentru a putea face setarile necesare asupra ei.
Tipul de textura cu care vom lucra deocamdata este GL_TEXTURE_2D
Deci instructiunea va fi:
glBindTexture(GL_TEXTURE_2D,id_textura);

Se vor seta apoi parametrii texturii:

1) Cum se comporta textura in cazul in care trebuie marita/micsorata pentru a se potrivi pe suprafata modelului.
   Proprietatile sunt:
   - GL_TEXTURE_MIN_FILTER
   - GL_TEXTURE_MAG_FILTER

   Valorile posibile sunt:
   - GL_LINEAR
   - GL_NEAREST
   - Pentru MIN mai sunt si niste proprietati care tin de mipmapping

   Exemplu:
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
2) Felul in care se comporta textura pentru coordonate u,v in afara intervalului [0,1].
   Proprietatile sunt:
   - GL_TEXTURE_WRAP_S (asociat coordonatei u)
   - GL_TEXTURE_WRAP_T (asociat coordonatei v)

   Valorile posibile sunt:
   - GL_CLAMP_TO_EDGE
   - GL_REPEAT
   - GL_MIRRORED_REPEAT

   Exemplu:
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);

Se creeaza imaginea (ca un fel de buffer) de tip textura cu functia:

glTexImage2D(tip_textura, nivel_mipmap, format_intern_textura, latime, inaltime, border=0, format_pixel_data, mod_memorare_pixel_data, pointer_vector_pixeli)

Parametrii sunt obtinuti din proprietatile fisierul textura calculate cu functia LoadTGA (din TGA.h, proiectul Utilities).

Exemplu:

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height , 0, GL_RGBA, GL_UNSIGNED_BYTE, array_pixeli);

Pentru desenare se va transmite catre shader un intreg de tip uniform, care va reprezenta indicele texturii. Cand avem o singura textura, acesta este 0, altfel ar trebui sa corespunda indicelui dat in functia *glActiveTexture*. Unui model ii pot fi atribuite mai multe texturi, caz in care vor trebui pe rand legate de indici (distincti, evident) de textura activa, pentru a putea fi folosite in shader.

Exemple:

glActiveTexture(GL_TEXTURE0); //nu e necesar acest apel daca avem o singura textura, deoarece implicit textura activa e GL_TEXTURE0.

glActiveTexture(GL_TEXTURE3);//in acest caz va trebui transmis 3 catre shader (vezi mai jos).

Dupa un apel glActiveTexture(GL_TEXTURE***i***), se realizeaza un glBindTexture(tip_textura, idt), pentru a lega textura cu id-ul *idt* de indicele de textura GL_TEXTURE***i***.

Apoi se transmite indicele de textura activa catre shader cu un glUniform1i:

glUniform1i(textureUniform,i);

In cazul in care avem o singura textura, instructiunea va fi:

glUniform1i(textureUniform,0);

Lucrul cu texturi in shader

In **vertex shader** trebuie doar sa ne asiguram ca se va realiza interpolarea coordonatelor uv:

v_uv=a_uv; // unde a_uv este un vec2, atributul corespunzator coordonatelor uv asociate vertexului, iar v_uv este un varying, tot vec2.

In **fragment shader** se va folosi variabila de tip uniform care identifica textura, insa cu un tip special de date: sampler2D:

uniform sampler2D u_texture;

Culoarea fragmentului va fi calculata cu ajutorul functiei *texture2D()*:

gl_FragColor = texture2D(u_texture, v_uv);