Računalna grafika: Laboratorijska vježba 2
OBJ format i osnove iscrtavanja trokuta

Toma Sikora[1]

FESB, Sveučilište u Splitu
toma.sikora@fesb.hr

1 Uvod

Druga vježba kolegija Računalna Grafika podijeljena je na dva dijela: na strukturu popularnog formata datoteke .obj[1] te uvod u iscrtavanje trokuta na zaslonu računala. Tokom ove vježbe radit ćemo na okolini postavljenoj u prošloj vježbi kako bi se koristili funkcijama specifikacije OpenGL, a za rad sa .obj datotekama koristit ćemo osnovne funkcije C/C++ programskog jezika.
Kao potporu implementaciji iscrtavanja algoritama u drugoj vježbi koristite se uputama sa stranice `https://learnopengl.com/Getting-started/Hello-Triangle`. Kako su upute poprilično opširne, nije bitno znanje svakog pojedinog koraka, već razumijevanje funkcija pojedinih grupa linija koda (npr. kako napraviti shader i prebaciti ga na grafičku karticu).

2 Format datoteke OBJ

Razvijen od strane tvrtke Wavefront Technologies[2], .obj format datoteke osmišljen je za pohranu 3D modela u računalima. Format je izrazito jednostavan i primarno namijenjen za pohranu podataka o 3D geometriji, kao što su pozicije vrhova trokuta, njihove teksture, definicije grupa vrhova koje čine poligone, normale poligona i slično. Wavefront Technologies je od početka držao format potpuno otvoren tako da su ga kroz godine preuzele i druge tvrtke koje se bave 3D grafikom. A kako su .obj datoteke pisane u ASCII formatu, lako su čitljive za ljude i za računala. Danas je .obj jedan od standarda za pohranu podataka o 3D geometriji koji većina ozbiljnih sustava za 3D grafiku podržava. Cilj prvog zadatka u vježbi biti će napisati kod za stvaranje i čitanje iz datoteke .obj.

2.1 Definicija formata

Struktura formata .obj funkcionira na sljedeći način. Dokument čine linije teksta, svaka od kojih predstavlja informaciju o 3D modelu označeno sa početnim znakovima linije, kao što su na primjer vrhovi v. Te su linije podijeljene u grupe ovisno o tipu informacije. Svaka linija koja počinje sa znakom # predstavlja komentar.

[1] https://docs.fileformat.com/3d/obj/
[2] http://www.wavefront.com/

Postoje brojne mogućnosti, a najčešće su sljedeće:

```
# Inicijalizacija nove grupe
# Ako jedan .obj file ima više grupa, svaka mora početi s
g [name]

# Lista vrhova geometrije
# Format zapisa koordinata je x y z [w] gdje je w opcionalno i default 1.0
v 0.123 0.234 0.345 1.0
v 0.321 0.432 0.543

# Lista poligona.
# Format zapisa trokuta je vertex_index/texture_index/normal_index
# Vrhovi se indeksiraju od 1, ne od 0
# Za sada koristimo samo prvu brojku za svaki vrh pa je dovoljan zapis:
f 1 2 3
# U kasnijim laboratorijima dolaze teksture i normalne:
f 3/1 4/2 5/3
f 6/4/1 3/5/3 7/6/5
f 7//1 8//2 9//3

# Lista normala
# Format zapisa vektora normale za svaki vrh je x y z
vn 0.707 0.000 0.707

# Lista tekstura
# Format zapisa koordinata je (u, [v, w])
# Vrijednosti su od 0 do 1, a v, w su opcionalni sa defaultom 0
vt 0.500 1 0
```

Osim ovih postoje i drugi tipovi informacije koje podrazumijeva format .obj, kao što su slobodne krivulje ili površine, ali njih nećemo koristiti u sklopu ovih vježbi.

2.2 Obavezna priprema

Prvi je zadatak ove vježbe napisati C/C++ kod za čitanje datoteke formata .obj (sami odaberite hoće li to biti funkcija kojoj se predaje datoteke i u kakvoj će strukturi bit zapisani pročitani podaci). Potrebno je učitavati **samo informacije o vrhovima** (npr. `v 0.5 0.0 0.2`) i **trokutima** (`f 3 6 7`). Rezultat izvršavanja programa treba biti otvorena i pročitana datoteka .obj (primjer je dan u materijalima vježbe). Što se tiče formata zapisa linija, koristite zapise `v x y z` za vrhove i `f v1 v2 v3` za trokute. Naravno, što je kod urednije napisan, lakše će ga biti kasnije nadograditi za teksture i normale.

3 Iscrtavanje trokuta na zaslonu računala

Drugi je zadatak ove vježbe **iscrtati trokute** s koordinatama iz pročitane datoteke. Iscrtavanje treba napraviti uz pomoć specifikacije OpenGL i GLFW biblioteke, kao što je objašnjeno u prošloj vježbi. Za preciznije upute o procesu iscrtavanja trokuta proučite stranice na learnopengl.com:

- `https://learnopengl.com/Getting-started/Hello-Triangle` i
- `https://learnopengl.com/code_viewer_gh.php?code=src/1.getting_started/2.2.hello_triangle_indexed/hello_triangle_indexed.cpp`

One sadrže detaljno opisan proces, a u nastavku je popis potrebnih naredbi na jednom mjestu.

3.1 Učitavanje liste vrhova i liste poligona

OpenGL nalaže kako se informacije o 3D modelu na ulazu u grafički pipeline učitavaju u obliku buffer objekata. Kroz njih, OpenGL može prebaciti veliku količinu podatak na grafičku karticu. Primjer naredbi potrebnih za učitavanje podataka o vrhovima i poligonima u memoriju grafičke kartice je sljedeći:

```
unsigned int VBO, VAO, EBO;
glGenVertexArrays(1, &VAO);
glGenBuffers(1, &VBO);
glGenBuffers(1, &EBO);
// bind the Vertex Array Object first, then bind and set vertex buffer(s), and then configur
glBindVertexArray(VAO);

glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);

glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);

glBindVertexArray(0);
```

3.2 Shader programi

Kako bi iscrtali poligon na zaslonu računala, koristit će se shader programi. Oni služe za definiciju malih poslova koje treba izvršiti na grafičkoj kartici i pisani su u GLSL programskom jeziku (u stilu programskog jezika C). Sljedeća vježba koncentrirat će se na pisanje svojih shadera prateći dobra načela u programiranju, a u ovoj vježbi koristit ćemo dva osnovna primjera shadera: vertex i fragment shader.

3.3 Korištenje shader programa u OpenGL-u

Sve bitne informacije o implementaciji i korištenju shadera dostupne su na `https://learnopengl.com/Getting-started/Hello-Triangle`, ali ukratko, najbitniji koraci su:

– učitavanje teksta GLSL koda za shader u *constchar∗* tip podataka:

```
const char *vertexShaderSource = "#version 330 core\n"
"layout (location = 0) in vec3 aPos;\n"
"void main()\n"
"{\n"
"   gl_Position = vec4(aPos.x, aPos.y, aPos.z, 1.0);\n"
"}\0";
```

– build i kompilacija shader programa:

```
// vertex shader
unsigned int vertexShader = glCreateShader(GL_VERTEX_SHADER);
```

```
glShaderSource(vertexShader, 1, &vertexShaderSource, NULL);
glCompileShader(vertexShader);
// check for shader compile errors
int success;
char infoLog[512];
glGetShaderiv(vertexShader, GL_COMPILE_STATUS, &success);
if (!success)
{
    glGetShaderInfoLog(vertexShader, 512, NULL, infoLog);
    std::cout << "ERROR::SHADER::VERTEX::COMPILATION_FAILED\n" << infoLog << std::endl;
}
```

– linkanje kompajliranih shader programa:

```
unsigned int shaderProgram = glCreateProgram();
// attach the shader programs one by one
glAttachShader(shaderProgram, vertexShader);
glAttachShader(shaderProgram, fragmentShader);
glLinkProgram(shaderProgram);
// check for linking errors
glGetProgramiv(shaderProgram, GL_LINK_STATUS, &success);
if (!success) {
    glGetProgramInfoLog(shaderProgram, 512, NULL, infoLog);
    std::cout << "ERROR::SHADER::PROGRAM::LINKING_FAILED\n" << infoLog << std::endl;
}
glDeleteShader(vertexShader);
glDeleteShader(fragmentShader);
```

Jednom kada smo definirali, kompajlirali i linkali shader programe, unutar glavne while rendering petlje main funkcije koristimo ih ovako:

```
glUseProgram(shaderProgram);
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

`GL_TRIANGLES` definira iscrtavanje trokuta kao osnovnog oblika, 0 definira indeks prvog čitanog vrha u buffer-u, a 3 definira broj vrhova koje prikazujemo iz buffer-a.

3.4 Obvezna priprema

Drugi zadatak obavezne pripreme je iscrtati trokut definiran .obj datotekom u materijalima na zaslon računala. Jednom kada je sve implementirano, program bi se trebao moći kompajlirati i pokrenuti, a rezultat iscrtavanja trebao bi izgledati slično slici 1 (naravno, ako su koordinate vrhova drugačije, trokut će biti drugačije postavljen).

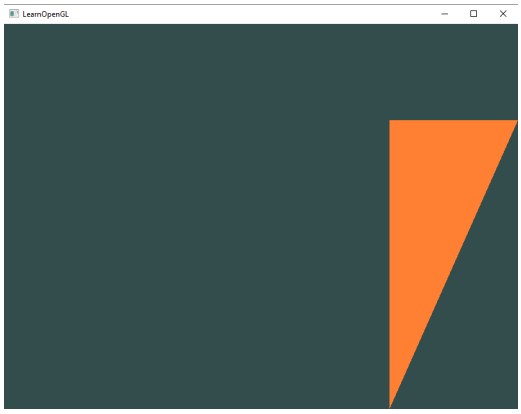

Fig. 1. Rezultat

4 Zaključak

U ovoj laboratorijskoj vježbi upoznali smo se s formatom datoteka .obj i osnovama iscrtavanja trokuta na zaslonu računala. Tokom idućih vježbi, to će nam služiti kao baza za implementaciju raznih algoritama naučenih u teoriji na predavanjima. Do sljedećeg puta, koga zanima može provjeriti sljedeće potencijalno korisne linkove:

1. Specifikacija GLSL jezika: `https://www.khronos.org/opengl/wiki/OpenGL_Shading_Language`
2. Definiranje materijala putem formata datoteka MTL: `https://paulbourke.net/dataformats/mtl/`
3. Dokumentacija OpenGL naredbi: `https://docs.gl/`
4. Na stranicama Merlina dostupni su i drugi primjeri .obj datoteka. Za testirati učitavanje i iscrtavanje trokuta, koga zanima može ih pokušati iscrtati.