Naloga 2 – Osnove OpenGL

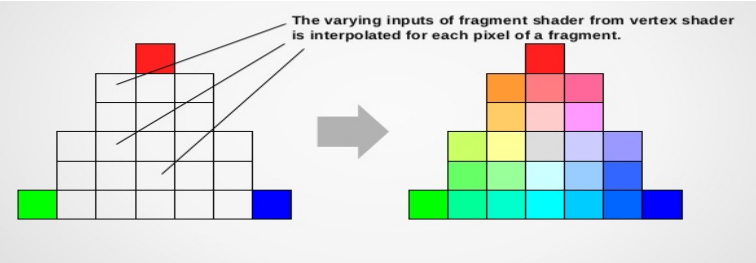
OpenGL je specifikacija, ki določa aplikacijski programski vmesnik za izris 3D grafike s pomočjo podpore za strojno pospeševanje na grafični procesni enoti. Implementacije OpenGL so izvedene v lastniških gonilnikih grafične kartice, obstajajo pa tudi odprto-kodne implementacije, ki omogočajo poganjanje tako na različnih grafičnih procesnih enotah kot tudi na centralni procesni enoti (npr. Mesa).

OpenGL je od različice 3 naprej doživel veliko sprememb, ki omogočajo bolj učinkovit izris. Geometrija za prikaz se nahaja znotraj pomnilnika na grafični kartici. Sam izris pa se v večji meri kontrolira s pomočjo programirljivih stopenj grafičnega cevovoda (senčilniki oglišč, fragmentov, geometrije, teselacije, izračuna).

Sodobne grafične kartice omogočajo programerju, da neposredno vpliva na postopek senčenja z definicijo senčilnikov (*shaders*), ki jih implementira v programskem jeziku GLSL (*GL shading language*) in se izvedejo na grafični procesni enoti. Za nas sta pomembna predvsem senčilnik oglišč (*vertex shader*) in senčilnik fragmentov (*fragment shader*). Kodo senčilnikov naložimo med samim izvajanjem aplikacije (*glCreateShader* in *glShaderSource*), jo prevedemo (*glCompileShader*), dodamo v senčilni program (*glAttachShader*), ki ga še predhodno ustvarimo (*glCreateProgram*). Program nato povežemo (*glLinkProgram*) in uporabimo pri samem izrisu (*glUseProgram*).

Senčilnik oglišč se kliče za vsako oglišče gradnikov, ki jih upodabljamo (npr. trikrat za vsak trikotnik). Vhodi v senčilnik oglišč so atributi (*attributes*) posameznega oglišča (npr. položaj, osnovna barva in ostali poljubni uporabniško definirani parametri), ki jih v kodi senčilnika deklariramo kot globalne spremenljivke z rezervirano besedo *in*. Vrednosti atributov so specifične za vsako posamezno oglišče. Poleg atributov senčilniku oglišč podamo tudi uniformne spremenljivke, ki se ne spreminjajo od oglišča do oglišča, ampak predstavljajo konstantne vrednosti za celotno senčeno geometrijo. Deklariramo jih z rezervirano besedo *uniform*, primeri pa so transformacijske matrike, barva itd.

Izhodi iz senčilnika oglišč so preračunane vrednosti atributov oglišča, kot so transformiran položaj, koordinate za teksture, barva, normala ali izračunana difuzna osvetlitev oglišča. Deklariramo jih z rezervirano besedo *out* in se v nadaljevanju grafičnega cevovoda interpolirajo za vse vmesne točke trikotnikov (Slika 1). Izhod iz senčilnika oglišč mora v vsakem primeru biti vsaj transformiran položaj oglišča, za katerega v ta namen obstaja vgrajena izhodna spremenljivka *gl_Position*, ki je ni potrebno posebej deklarirati. Interpolirane vrednosti postanejo vhodi v senčilnik fragmentov, ki mora imeti iste spremenljivke deklarirane z rezervirano besedo *in*, poleg njih pa lahko prav tako uporablja uniformne spremenljivke (deklarirane z *uniform*) in tudi teksture (deklarirane s *sampler2D*).



Slika 1: Primer interpolacije. Vrednosti (barve) fragmentov se interpolirajo glede na oddaljenost od pripadajočih oglišč trikotnika *(http://www.slideshare.net/champ_yen/opengl-es-2x-programming-introduction)*

Pred samim prikazom geometrijskih gradnikov, je potrebno le te predhodno naložiti na grafično kartico, kjer lahko vrednosti atributov oglišč zapišemo v ločenih poljih (lahko pa tudi v prepletenem načinu v enem samem polju) in jih nato z *glBufferData* prepišemo v t.i. objekte predpomnilnika oglišč (*vertex buffer objects VBO*), ki so shranjeni na grafični kartici. Le te je potrebno predhodno eksplicitno ustvariti (*glGenBuffers*) in priklopiti (*glBindBuffer*).

Izris geometrije pomeni zagon grafičnega cevovoda, ki atribute oglišč pobira iz *VBO* in jih prireja vhodnim spremenljivkam senčilnika oglišč na podlagi povezav, ki jih ustvarimo preko generičnih indeksov atributov. Te povezave ustvarimo tako, da z *glBindAttribLocation* ali (*layout(location=X)* v senčilnem programu) posamezni vhodni spremenljivki senčilnika oglišč priredimo generični indeks, na katerega potem vežemo *VBO* z ukazom *glVertexAttribPointer*. Vse potrebne *VBO* je potrebno pred tem povezati z *glBindBuffer* v objekt polja oglišč (vertex array object – VAO) in jih aktivirati z *glEnableVertexAttribArray*.

Vrednosti uniformnim spremenljivkam določimo tako, da najprej v povezanem programu pridobimo njihov naslov (*glGetUniformLocation*), nato pa v odvisnosti od tipa spremenljivke z ustreznim ukazom (*glUniform*, *glUniformMatrix*) na ta naslov kopiramo vrednost. Upodabljanje sprožimo s priklopom *VAO* v grafični cevovod (*glBindVertexArray*) in klicem izrisa (*glDrawArrays*).

1. Zahteve naloge

Implementirajte grafično aplikacijo (najlažje s pomočjo predloge iz sistema za vaje), ki z uporabo OpenGL 3/4 (Core profile) prikaže piramido iz barvnih kock (za vsako kocko obvezno uporabite drugačno **barvo**). Za lažjo prostorsko predstavo izrišite tudi vodoraven štirikotnik, ki naj predstavlja tla. Izris mora biti izveden brez funkcij fiksnega cevovoda, ki se je uporabljal nekoč v prejšnjem tisočletju. Ukazi, kot so *glBegin*, *glEnd*, *glLight**, *glRotate*, *glScale, glLoadMatrix* torej **niso** dovoljeni.

Implementacijo lahko izvedete v programskem jeziku C++, C# ali Java, pri čemer lahko uporabite ustrezne knjižnice (npr. Qt, OpenTK, JOGL) za izvedbo preprostega uporabniškega vmesnika in sprejemanje uporabnikovih akcij (navadno okno, zajemanje tipk).

Postopek implementacije in točkovanje:

- Izrišite eno kocko : 2 točki
  - Napolnite VBO z geometrijskimi podatki ene kocke.
- Nato izrišite tla pod kocko: 1 točka,
  - V obstoječ VBO lahko dodate nove geometrijske podatke ploskve ali pa ustvarite nov VBO in VAO. Če nove geometrijske podatke dodate v obstoječ VBO, v funkciji glDrawArrays določite indeks začetnega oglišča (kjer se prične oglišče za tla).
- Izris vsaj treh kock z različnimi barvami: 1 točka.
  - V obstoječ VBO dodajte geometrijske podatke preostalih kock ali za vsako kocko ustvarite nov VBO in VAO. Vsako kocko izrišite z ločenim klicem glDrawArrays, kjer preko glUniform4f*, določite barvo kocke.