Interaktivna računalna grafika - dokumentacija

Matija Šantl

9. lipnja 2013.

Sadržaj

Poglavlje 1

1. laboratorijska vježba

Prvu laboratorijsku vježbu sam radio prema B uputama za izradu laboratorijskih vježbi. Kao jezik ostvarenja rješenja danih zadataka odabrao sam *C++*. Vrijeme uloženo u ostvarenje ove laboratorijske vježbe je oko 10 sati.

1.1 Zadatak 1. - Izrada pomoćne biblioteke

1.1.1 Kratak opis programa

Prvi zadatak s kojim smo se susreli na laboratorijskim vježbama je bila izrada pomoćne biblioteke. Pomoćna biblioteka se sastoji od više razreda koji nam omogućavaju lakši programski zapis izračuna operacija s vektorima i matricama.

Razredi koji su ostvareni u sklopu ove vježbu su *IVector*, *AbstractVector*, *Vector*, *IMatrix*, *MatrixTransposeView*, *MatrixSubmatrixView*, *AbstractMatrix*, *Matrix*, *MatrixVectorView*, *VectorMatrixView*.

Svaki je razred ostvaren u vlastitoj datoteci zaglavlja, te je za funkcionalnost pojedinog razreda potrebno samo uključiti odgovarajuću datoteku zaglavlja.

1.1.2 Korištene strukutre podataka

Strukture podataka koje su korištene u ovom zadatku su ujedno i sam zadatak kojeg je trebalo ostvariti.

1.1.3 Upute za korištenje programa

Nakon ostvarenja svih potrebna razreda, za provjeru ispravnosti istih, napisan je program koji na nekoliko primjera pokazuje rezultate izračuna unutar područja linearne algebre. Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos1* koju možemo pokrenuti i uvjeriti se u ispravnost rada ostvarenih razreda.

1.1.4 Komentar rezultata

Ovdje razvijena biblioteka je potrebna u daljnjim vježbama te je veoma važno da ona radi ispravno. Rezultati dobiveni pomoću razvijene biblioteke uspoređeni su s onima koji su ručno izrađeni te nije primjećena nepravilnost.

1.2 Zadatak 2. - Prvi program u OpenGL-u

1.2.1 Kratak opis programa

Prvi program u OpenGL-u je bio crtanje većeg broja ispunjenih trokuta u odabranoj boji. Program treba pamtiti trnutno aktivnu boju i prikazivati je u gornjem desnom uglu. Zadavanje trokuta u programu se obavlja mišem. Prilikom crtanja trokuta na ekranu, trokuti se crtaju redosljedom kojim su dodavni u listu.

1.2.2 Promjene načinjene s obzirom na upute

U uputi je zadan program u OpenGL-u kojeg je trebalo modificirati kako bismo ostvarili dani zadatak. Promjene koje su načinjene su u skladu s tekstom zadatka.

1.2.3 Korištene strukutre podataka

Strukture podataka koje su korištene za ostvarenje zadatka su prilagođene potrebama ovog zadataka, tako se za listu u koju spremamo aktivne trokute koristi strkutura podataka *vector* iz *STL* biblioteke. Dodatne strukture se koriste za spremanje trenutnog stanja programa, koja je boja iscrtavanja, točke trokuta kojeg crtamo itd.

1.2.4 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos2*. Pokretanjem izvršne datoteke otvara se prozor u kojem možemo crtati trokute u nekoj od ponuđenih boja. Zadavanje točaka se vrši pritiskom ljeve tipke miša, te se nakon zadavanja treće točke u trokut spremi u listu zadanih trokuta. Zadavanjem manje od tri točke, pomicanjem miša, iscrtava se ili linija ili trokut koja nam prikazuje kako će izgledati stanje ekrana ako u tom trenutku pritisnemo tipku miša.

1.2.5 Komentar rezultata

Kao rezultat pokretanja ovog progama možemo crtati trokute u nekoj od ponuđenih boja koje mijenjamo pritiskom na tipke p i n. Takav program nije kompliciran za ostvariti a stekne se dobar uvid u OpenGL što je potrebno za daljnje vježbe.

1.3 Zadatak 3. - Crtanje linija na rasterskim prikaznim jedinicama

1.3.1 Kratak opis programa

Koristeći OpenGL, treba korisniku omogućiti crtanje prozivoljnog broja linija. Korisnik linije zadaje mišem, na način da prvi klik definira početak segmenta a drugi klik kraj segmenta. Dodatno, korisnik može odabrati iscrtavanje kontrolnih linija gotovim funkcijama iz OpenGL-a i/ili aktivirati algoritam odsijecanja linija Cohen Sutherlanda.

1.3.2 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su *vector* iz *STL* biblioteke te razredi koji nam olakšavaju pamćenje segmenata koje trebamo crtati.

1.3.3 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos3*. Prilikom pokretanja izvršne datoteke otvara se prozor u kojem možemo raditi sljedeće. Klikom miša određujemo rubne točke segmenata koje želimo iscrtati. Tipkom *k* aktiviramo iscrtavanje kontrolne linije koja se iscrtava blago desno u drugoj boji. Tipkom *o* aktiviramo algoritam odsijecanja Cohen Sutherlanda, te se iscrtavaju segmenti koji su dio središnjeg prozora.

1.3.4 Komentar rezultata

U ovom zadatku je potrebno ostvariti Bresenhamov algoritam crtanja linija. Proučavajući taj algoritam vidimo koliko je važno da iskoristimo rastersku prirodu računalnih ekrana radi bržeg iscrtavanja. Zanimljivo je vidjeti i pametnu primjenu binarnog kodiranja segmenata tako da se pomoću binarnih operacija lako odredi položaj segmenta s obzirom na prozor u algoritmu Cohen Sutherlanda.

Poglavlje 2

2. laboratorijska vježba

Drugu laboratorijsku vježbu sam radio prema B uputama za izradu laboratorijskih vježbi. Kao jezik ostvarenja rješenja danih zadataka odabrao sam *C++*. Vrijeme uloženo u ostvarenje ove laboratorijske vježbe je oko 6 sati.

2.1 Zadatak 4. - Crtanje i popunjavanje poligona

2.1.1 Kratak opis programa

Koristeći OpenGL za crtanje, program treba korisniku omogućiti da mišem definira vrhove poligona i koji će potom korisniku prikazati taj poligon, omogućiti mu da dobije prikaz popunjenog poligona, te omogućiti korisniku da mišem zadaje točke za koje će program u konzolu ispisivati u kakvom je odnosu točka i poligon.

2.1.2 Promjene načinjene s obzirom na upute

Promjene načinjene s obzirom na upute su vezane uz prijedlog izračuna koeficijenata pravca bridova. Postignut je isti učinak uzimajući indekse pomaknute za jedan.

2.1.3 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su *vector* iz *STL* biblioteke te razredi koji nam olakšavaju rad s poligonima i provjeru odnosa točke i poligona.

2.1.4 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezul-

tat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos4*. Nakon pokretanja izvršne datoteke otvara nam se prozor u kojem možemo zadavati točke poligona. Zadani poligon tako može biti konveksan ili konkavan. S obzirom na izbor točaka, algoritam punjenja će *dobro* ispuniti poligon ako je on konveksan, inače će se primjetiti neke anomalije. Tipkom k mijenjamo vrijednost zastavice konveksnost koja nam omogućava, odnosno onemogućava zadavanje točaka poligona koje bi narušile njegovu konveksnost. Tipkom p mijenjamo vrijednost zastavice popunjavanje koja koristeći algoritam za popunjavanje poligona ispuni poligon zadanom bojom. Tipkom n radimo ciklički prelazak na sljedeće stanje. Ako smo u stanju 1 omogućeno nam je zadavanje točaka poligona, dok nam je u stanju 2 omogućeno zadavanje ispitnih točaka za koje se provjerava odnos točke i poligona.

2.1.5 Komentar rezultata

U zadatku je specificiran algoritam koji se koristi za popunjavanje poligona, te se prilikom zadavanja konkavnog poligona mogu primijetiti određene anomalije koje se kod konveksnih poligona ne javljaju. Ispitivanje odnosa točke i poligona također ovisi o prirodi zadanog poligona. Naime, ako je poligon konveksan, ispisivat će se dobri rezultati, ali ako je on konkavan, neki od rezultata će biti pogrešni.

2.2 Zadatak 5. - 3D tijela

2.2.1 Kratak opis programa

Program treba pročitati sadržaj *.obj* datoteke i u memoriju učitati definirani model tijela. Za svaki trokut treba izračunati i zapmatiti pripadne koeficijente jednadžbe ravnine. Porgram treba potom korisniku omogućiti da interaktivno unosi koordinate točaka u 3D prostoru, te nakon svake unesene točke program treba provjerit iu kakvom su odnosu unesena točka i tijelo te rezultat ispitivanja ispisati na ekran. Također, ako korisnik unese naredbu *normiraj*, na zaslon se u *.obj* formatu ispiše normirani model objekta.

2.2.2 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su *vector* iz *STL* biblioteke te razred *ObjectModel* koji nam služi za spremanje učitanog modela u *.obj* formatu. Razred *ObjectModel* koristi dva pomoćna razreda, *Vertex3D* i *Face3D*.

2.2.3 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos5*. Pokretanje pro-

grama se vrši s argumentom *.obj* datotekom iz koje učitavamo model. Nakon što se model učitao, imamo dvije mogućnosti. Prva je da unosom naredbe *normiraj* na zaslon dobijemo normirani model tijela, a druga je da unesemo točku iz 3D prostora, te na zaslon dobijemo rezultat odnosa točke i tijela (unutar, izvan ili na rubu).

2.2.4 Komentar rezultata

Ovaj zadatak je dan kao uvod za rad s 3D tijelima. Rezultati ispisa ovise o prirodi modela 3D tijela. Naime, ako je tijelo konkavno, može se dogoditi za neku točku program ispiše pogrešan odnos točke i tijela. Ako je tijelo konveksno, do toga ne bi smijelo doći.

Poglavlje 3

3. laboratorijska vježba

Treću laboratorijsku vježbu sam radio prema A uputama za izradu laboratorijskih vježbi. Kao jezik ostvarenja rješenja danih zadataka odabrao sam *C++*. Vrijeme uloženo u ostvarenje ove laboratorijske vježbe je oko 4 sata.

3.1 Zadatak 6. - Transformacija pogleda i perspektivna projekcija

3.1.1 Kratak opis programa

Program kao ulaz prima model 3D tijela u *.obj* formatu. Nad zadanim je modelom potrebno načiniti transformaciju pogleda i perspektivnu projekciju te iscrtati dobiveni poligon.

3.1.2 Korištene strukutre podataka

Strukture podataka koje su korištene u ovom zadatku su *vector* iz *STL* biblioteke, razred *Matrix* sa pripadnim metodama radi lakšeg zapisa prilikom računanja transformacija i projekcija.

3.1.3 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos6*. Program kao argumente naredbenog retka prima dvije datoteke. Prva je *.obj* formatirana datoteka u kojoj je opisan 3D model tijela dok druga datoteka sadrži dvije točke, točku očišta i točku gledišta. Nakon pokretanja programa sa danim nazivima datoteka, u otovrenom se prozoru iscrta poligon koji nastaje transformacijom pogleda i perspektivnom projekcijom zadanog 3D modela tijela.

3.1.4 Komentar rezultata

Rezultat ovog zadatka je poligon koji nastaje primjenom transformacije pogleda i perspektivne projekcije zadanog 3D modela tijela. Prilikom zadavanja točke očišta moramo paziti da je ne zadamo u unutrašnjosti objekta.

3.2 Zadatak 7. - Krivulja Bezijera

3.2.1 Kratak opis programa

Zadatak 7 je proširenje prethodnog zadataka u pogledu što točka očišta nije ista kad jednom pokrenemo program, već ju možemo mijenjati tijekom izvođenja programa. Točka očišta se kreće po zadanoj Bezierovoj krivulji koja se pak određuje iz točaka koje zadamo prilikom pokretanja programa.

3.2.2 Korištene strukutre podataka

U zadatku 7 su korištene iste strukture podataka kao i u prethodnom zadatku.

3.2.3 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos7*. Program uz ulaze iz prethodnog zadatka prima i točke iz kojih se računa Bezierova krivulja. Nakon pokretanja programa sa danima nazivima datoteka, u otvorenom se prozoru iscrtava poligon koji nastaje transformacijom pogleda i perspektivnom projekcijom zadanog 3D modela tijela s obizrom na točku očišta. Točku očišta pomičemo po Bezierovoj krivulji pritiskom na tipku j kako bismo povećali parametar t ili pritiskom na tipku k kako bismo smanjili parametar t.

3.2.4 Komentar rezultata

Rezultat je sličan onom dobivenom u prethodnom zadatku samo što pritiskom na tipke t i k možemo pomicati položaj očišta a time i izgled iscrtanog poligona.

Poglavlje 4

4. laboratorijska vježba

Četvrtu laboratorijsku vježbu sam radio prema A uputama za izradu laboratorijskih vježbi. Kao jezik ostvarenja rješenja danih zadataka odabrao sam *C++*. Vrijeme uloženo u ostvarenje ove laboratorijske vježbe je oko 4 sata.

4.1 Zadatak 8. - Sjenčanje tijela

4.1.1 Kratak opis programa

Potrebno je učitati 3D model tijela i uz različite načine sjenčanja i uklanjanje stražnjih poligona iscrtati dobivenu sliku. Za potrebe transformacije pogleda i perspektivne projekcije koristimo OpenGL funkcije.

4.1.2 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su *vector* iz *STL* biblioteke te razredi u koje spremamo točke i poligone.

4.1.3 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos8*. Program kao argument prima naziv datoteke koja sadrži model 3D modela tijela u *.obj* formatu. Nakon pokretanja programa prikazuje se žičana forma danog 3D modela uz aktiviarano uklanjanje stražnjih poligona. Pritiskom na tipku *p* mijenjano način prikaza. Dostupni načini prikaza su prikaz tijela uz konstantno sjenčanje i prikaz tijela uz Gouraudovo sjenčanje. Pritiskom na tipku *l* pomičemo očište po x-osi za 0.1. Pritiskom na tipku *h* pomičemo očište po x-osi za -0.1. Pritiskom na tipku *j* pomičemo očište po y-osi za 0.1. Pritiskom na tipku *k* pomičemo očište po y-osi za -0.1.

4.1.4 Komentar rezultata

Program sadrži više načina prikaza te je time i dosta efektan. Žičana forma nam daje dobar prikaz o radu algoritma uklanjanje stražnjih poligona. Usporedba konstatnog i Gouraudovog sječanja pak pokazuje razliku koju donosi malo složeniji račun na prikaz osvjetljenja tijela.

4.2 Zadatak 9. - Fraktali

4.2.1 Kratak opis programa

Potrebno je nacrtati Mandelbrotov i Julijev skup za točke kompleksne ravnine koje odgovaraju točkama prozora u kojem crtamo.

4.2.2 Promjene načinjene s obzirom na upute

Kako je zadatak 9. preuzet iz A verzije laboratorijskih vježbi, funkcija koja određuje boju na temelju brzine divergencije je preuzeta iz B verzije laboratorijskih vježbi.

4.2.3 Korištene strukutre podataka

Korištena je struktura podataka koja predstavlja kompleksne brojeve radi lakšeg zapisa operacija na kompleksim brojevima u programu.

4.2.4 Upute za korištenje programa

Za potrebe kompilacije izvornog koda, napisana je *Makefile* datoteka koja automatizira proces kompilacije više datoteka u jednu izvršnu datoteku. Kao rezultat rada *Makefile* datoteke dobijemo izvršnu datoteku *labos9*. Nakon pokretanja programa od korisnika se zahtjeva da unese parametre koji određuju fraktale. Nakon toga se prikazuje Mandelbrotov skup te se pritiskom na tipku p možemo prebacivati između prikaza Julijevog i Mandelbrotovg skupa.

4.2.5 Komentar rezultata

Program iscrtava fraktale, matematičke umotvorine. Umjesto dosadnog crnobijelog bojanja u ovisnosti o konvergenciji, odnosno divergenciji točke korišteno je bojanje koje u obzir uzima brzinu divergencije te je time postignut efektniji prikaz.