Interaktivna računalna grafika Labosi - dokumentacija

Mato Manović 0036484071

31. svibnja 2017.

Sadržaj

Napomena: Prve dvije laboratorijske vježbe su izrađene prema uputama za B inačicu laboratorijskih vježbi iz predmeta Interaktivna računalna grafika, dok su preostale dvije izrađene prema A inačici uputa. U ovom slučaju prvih pet zadataka pripadaju prvim dvjema laboratorijskim vježbama, dok preostala četiri zadatka u ovoj dokumentaciji 6,7,8 i 9 su u originalnoj uputi za A inačicu 5,6,7 i 8 zadatak.

1 Prva laboratorijska vježba

1.1 Zadatak 1. - Izrada pomoćne biblioteke

1.1.1 Kratak opis programa

U prvom zadatku prve laboratorijske vježbe bilo je potrebno izraditi vlastitu matematičku biblioteku koju ćemo koristiti za izradu sljedećih laboratorijskih vježbi.Biblioteka treba nuditi funkcionalnosti rada s osnovnim operacijama u linearnoj algebri to jest ponuditi sve osnovne operacije nad vektorima i matricama.U tu svrhu su modeliran je osnovni razred *Matrix* i *Vector* kao posebni slušaj matrice koja ima jedan red ili stupac, ali je izdvojena kao poseban razred budući da su najčešće operacije u interaktivnoj računalnoj grafici nad vektorima.

Biblioteka je napisana u programskog jeziku C++ koji je korišten i za izradu ostalih laboratorijskih vježbi.Razredi koji su ostvareni u ovoj laboratoriskoj vježbi su IVector, AbstractVector, Vector, IMatrix, MatrixTransposeView, MatrixSubmatrixView, AbstractMatrix, Matrix, MatrixVectorView, VectorMatrixView.Svaki razred je radi preglednosti i u skladu s načelima oblikovanja programskog koda izdvojen u posebnu datoteku.

1.1.2 Korištene strukutre podataka

Osnovna struktura podataka koja je enkapsulirana i koja služi za pohranu elemenata razreda Vector i Matrix je razred vector iz standardne biblioteke C++(STL).

1.1.3 Upute za korištenje programa

Cijeli projekt je napravljen korištenjem CodeBlocks IDE-a za razvoj aplikacija u programskom jeziku C i C++. Stoga je dovoljno CodeBlocks-u otvoriti projekt i pokrenuti razred koji sadrži glavnu metodu main. U njoj su napisani nekoliko primjera koji demonstriraju i provjeravaju ispravnost rada svih izrađenih 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 su rezultati u skladu s očekivanim.

1.2 Zadatak 2. - Prvi program u OpenGL-u

1.2.1 Kratak opis programa

Zadatak je bio ostvariti crtanje nekoliko trokuta ispunjenih sa zadanom bojom.U uglu ekrana se nalazio mali kvadratić ispunjen trenutno aktivnom bojom koju koristimo za iscrtavanja trokuta.Zadavanja trokuta obavljamo klikovima mišem po ekranu pri čemu nakon što zadamo dvije točke možemo vidjeti u pokretu trenutni izgled trokuta prije nego što konačno potvrdimo izbor treće točke trokuta.Trokuti se iscrtavaju redoslijedom kojim su dodani tako da će oni trokuti koji su zadnji dodani biti vidljivi.

1.2.2 Promjene načinjene s obzirom na upute

Što se tiče samih uputa nisu učinjene nikakve promjene, npr. u uputama nisu eksplicitno navedene tipke koje trebamo koristiti za promjenu trenutno aktivne boje što je prepušteno slobodnoj volji.

1.2.3 Korištene strukture podataka

Strukture podataka koje su korištene za ostvarenje zadatka su prilagođene potrebama ovog zadataka, tako se za listu u koju spremamo aktivne trokute koristi struktura podataka vector iz standardne biblioteke C++-a. Dodatne strukture koje se koriste potrebne su za spremanje trenutnog stanja programa, koja je boja iscrtavanja, točke trokuta kojeg crtamo i tako dalje.

1.2.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju program ne prima nikakve ulazne parametre.

1.2.5 Komentar rezultata

Kao rezultat pokretanja ovog programa možemo crtati trokute u nekoj od ponuđenih boja koje mijenjamo pritiskom na tipke p(previous) i n(next). Namjera ovog programa je bilo stjecanje osnovnog znanja potrebnog za rad s OpenGl-om, a to je postignuti jednostavnim primjerom kao što je crtanje trokuta u kojoj imamo interakciju korisnika sa programskim sučeljem.

1.3 Zadatak 3. - Crtanje linija na rasterskim prikaznim jedinicama

1.3.1 Kratak opis programa

Koristeći OpenGL, trebalo je korisniku omogućiti crtanje proizvoljnog broja linija. Korisnik linije zadaje mišem, na način da prvi klik definira početak segmenta a drugi klik kraj segmenta. Dodatno, korisnik ima mogućnost odabira načina iscrtavanja kontrolnih linija gotovim funkcijama iz OpenGL-a ili iscrtavanje linija pri čemu se isključivo koristi Bresenhamov algoritam implementiran u ovom zadatku. Također korisniku je na raspolaganju opcija prikaza manjeg dijela prozora koji jedini prikazuje linije, dok ostatak linija i linije koje nisu u aktiviranom području nisu iscrtavane.To je ostvareno algoritom Cohen-Sutherland.

1.3.2 Promjene načinjene s obzirom na upute

U ovom slučaju nije bilo potrebne raditi nikakve posebne preinake budući da su većina stvari bile eksplicitno definirane pa čak i tipke potrebne za aktivaciju pojedinih algoritama.

1.3.3 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su vector iz standardne biblioteke C++-a te razredi koji nam olakšavaju pamćenje segmenata koje trebamo crtati.

1.3.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju program ne prima nikakve ulazne parametre. 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.

1.3.5 Komentar rezultata

U ovom zadatku je bilo potrebno ostvariti Bresenhamov algoritam crtanja linija. Pomoću njega smo dobili uvid u nemogućnost točnog prikazivanja nekog elementa kao što je linija zbog diskretne podjele prikazne jedinice koju nazivamo rasterska jedinica. Također je bilo potrebno implementirati algoritam Cohen-Sutherland koji je vrlo zanimljiv zbog

kodiranja područja s određenim binarnim kodom i binarnim operacijama među njima koje su vrlo brze i učinkovite.

2 Druga laboratorijska vježba

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

U ovom slučaju nije bilo potrebne raditi nikakve posebne preinake budući da su većina stvari bile eksplicitno definirane pa čak i tipke potrebne za promjenu pojedinih stanja.

2.1.3 Korištene strukture 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

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju program ne prima nikakve ulazne parametre.Nakon pokretanja izvršne datoteke otvara se prozor u kojem zadajemo točke poligona. Poligoni koje iscrtavamo mogu biti konveksni i konkavni. 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.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.Sam algoritam koristi definiciju lijevih i desnih bridova te ukoliko je poligon konkavan nećemo imati ispravne vrijednosti lijeve i desne koordinate horizontalne linije koje iscrtavamo na rasteru. Ispitivanje odnosa točke i poligona također ovisi o prirodi zadanog poligona. Naime, ako je poligon konveksan, ispisivat će se dobri rezultati, ali u slučaju konkavnosti rezultat algoritma je nedefiniran to jest neće davati valjanu vrijednost.

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 zapamtiti pripadne koeficijente jednadžbe ravnine. Program treba korisniku omogućiti da interaktivno preko konzole unosi koordinate točaka u 3D prostoru, te nakon svake unesene točke program treba provjerit u 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 Promjene načinjene s obzirom na upute

U ovom slučaju nije bilo potrebne raditi nikakve posebne preinake budući da su većina stvari bile eksplicitno definirane među kojima su i preporučeni nazivi i strukture podataka potrebni za implementaciju.

2.2.3 Korištene strukture podataka

Strukture koje su korištene za ostvarenje ovog zadatka su vector iz standardne biblioteke C++-a te razred ObjectModel koji nam služi za spremanje učitanog modela u .obj formatu. Razred ObjectModel koristi dva pomoćna razreda, Vertex3D koji predstavlja točku i Face3D koji predstavlja ravninu.

2.2.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom

slučaju program prima jedan argument putanju do .obj datoteke u kojoj je pohranjen model tijela. Nakon pokretanja programa u konzoli zadajemo koordinate točke u trodimenzionalnom prostoru pri čemu dobivamo ispis je li ta točka unutar ili izvan učitanog tijela.Također dostupna je naredba normiraj koja ispisuje normirani model tijela gdje maksimalni raspon koordinate u granicama od -1 do 1.Rad programa zaustavljamo naredbom exit.

2.2.5 Komentar rezultata

Ovaj zadatak je dan kao uvod za rad s 3D tijelima. U ovom zadatku slično kao i u prethodnom algoritam za odnos točke i tijela u slučajevima nekonveksnih tijela daje nedefiniran rezultat to jest nije valjan kao u slučaju konkveksnog tijela. U ovom zadatku se upoznajemo i s normalizacijom tijela koja može poslužiti kasnije za lakše iscrtavanje tijela na rasterskoj jedinici.

3 Treća laboratorijska vježba

3.1 Zadatak 6. - Transformacija pogleda i perspektivna projekcija

3.1.1 Kratak opis programa

Program kao ulaz prima model 3D tijela .obj formatu. Nad zadanim je modelom potrebno načiniti transformaciju pogleda i perspektivnu projekciju te iscrtati dobiveno tijelo nakon što su primijenjene zadane trasnformacije.

3.1.2 Promjene načinjene s obzirom na upute

U ovom zadatku također nije bilo potrebne raditi nikakve posebne preinake budući da su većina stvari bile eksplicitno definirane među kojima su i formule za računanje matrica i upute za njihovo korištenje pri transformacijama.

3.1.3 Korištene strukture podataka

Strukture koje su korištene za ostvarenje ovog zadatka su vector iz STL biblioteke te razredi koji predstavljaju vrh i ravninu, to jest korištene su identične strukture podataka koje su korištene u prethodnom zadatku za pohranjivanje modela objekta.

3.1.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju zadajemo dva parametra putanju do prve datoteke koja predstavlja model tijela kojeg učitavamo u .obj formatu te drugi argument koji predstavlja putanju do datoteke koja pohranjuje koordinate točke očista i gledišta.Nakon pokretanja programa u otvorenom prozoru dobivamo 2D iscrtano tijelo koje nastaje primjenom transformacije pogleda i perspektivnom projekcijom učitanog tijela.

3.1.5 Komentar rezultata

Rezultat ovog zadatka je poligon koji nastaje primjenom transformacije pogleda i perspektivne projekcije zadanog 3D modela tijela. Prilikom zadavanja točke očišta trebamo pripaziti da je ne zadamo u unutrašnjosti objekta jer tada nećemo jasno vidjeti objekt kojeg iscrtavamo. Također dimenzija slike ne ovisi o udaljenosti očišta i gledišta to jest koordinate točaka od perspektivne projekcije su normirane i veličina projekcije odgovara približno veličini ekrana.

3.2 Zadatak 7. - Krivulja Bezijera

3.2.1 Kratak opis programa

Ovaj zadatak je proširenje prethodnog zadatka u smislu da koordinatu očišta ili gledišta možemo pomicati po zadanoj Bezierovoj krivulji. U ovom zadatku također učitavamo tijelo u .obj formatu i radimo iste transformacije pogleda i istu perspektivnu projekciju kao u prethodnom zadatku.

3.2.2 Promjene načinjene s obzirom na upute

U ovom slučaju nije skroz eksplicitno definirano koju točku pomičemo po zadanoj Bezierovoj krivulji stoga sam u ovom slučaju odabrao točku očišta, ali konceptualno je slična stvar. Također implementacija je razdvojena u dva dijela tj. dva zasebna programa. Prvi program crta zadanu Bezierovu krivulju na zaslon. Dok drugi program je spoj prethodnog zadatka i programa za crtanje Bezierove krivulje. Također nisu eksplicitno definirane tipke koje koristimo za kretanje očišta što je prepušteno slobodnoj volji.

3.2.3 Korištene strukture podataka

Strukture koje su korištene za ostvarenje ovog zadatka su vector iz STL biblioteke te razredi koji predstavljaju vrh i ravninu, to jest korištene su identične strukture podataka koje su korištene u prethodnom zadatku za pohranjivanje modela objekta.

3.2.4 Upute za korištenje programa

Sam program je napisan je u dvije .cpp datoteke bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.Za prvi program u naredbenom retku zadajemo putanju do datoteke u kojoj su zadane koordinate kontrolnog poligona Bezierove krivulje. Na osnovu tih podataka iscrtavamo krivulju sa korakom parametra t u iznosu 0.01 što znači da krivulju aproksimiramo s približno sto

povezanih točaka.Drugi program prima dva argumenta gdje prvi argument predstavlja putanju do .obj datoteke u kojoj je pohranjen model tijela te drugi argument koji predstavlja putanju do datoteke koja predstavlja Bezierovu krivulju po kojoj ćemo gibati točku očišta. Točku očišta pomičemo lijevo-desno pritiskom na tipke s i d.Pritiskom na tipku p možemo ukloniti stražnje poligone tijela.

3.2.5 Komentar rezultata

Rezultat je sličan onom dobivenom u prethodnom zadatku samo što pritiskom na tipke s i d možemo pomicati položaj očišta,a time i izgled iscrtanog poligona.Ovo znači da svakim pomicanjem točke moramo ponovo raditi složeni izračun transformacije pogotovo u slučaju kada ne uključimo algoritam za uklanjanje stražnjih poligona. Ipak u ovim slučajevima i najzahtjevniji modeli nisu stvarali probleme.

4 Četvrta laboratorijska vježba

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 Promjene načinjene s obzirom na upute

U ovoj implementaciji nisu načinjene nikakve promjene vezano uz samu osnovnu ideju zadatke to jest dodane su funkcije koje korisniku omogućavaju pomicanje koordinate točke očišta i koordinate točke gledišta. Također pritiskom na tipku p mijenjamo trenutni prikaz kojih ima tri vrste to jest prikaz žičane strukture, prikaz tijela s konstantnim sjenčanjem i prikaz tijela s Gouraudovim sjenčanjem.

4.1.3 Korištene strukutre podataka

Strukture koje su korištene za ostvarenje ovog zadatka su vector iz STL biblioteke te razredi koji predstavljaju vrh i ravninu, to jest korištene su identične strukture podataka koje su korištene u prethodnom zadatku za pohranjivanje modela objekta.

4.1.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju zadajemo jedan argument koji predstavlja putanju do .obj datoteke koja predstavlja model tijela kojeg učitavamo. Zatim u konzoli još moramo zadati početnu koordinatu očišta i početnu koordinatu izvora svjetlosti.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. Također pritiskom na tipku r možemo vratiti prikaz u početno stanje.

4.1.5 Komentar rezultata

Program sadrži više načina prikaza te je time i dosta efektan. Žičana forma nam daje dobar uvid o radu algoritma uklanjanje stražnjih poligona i njegovu valjanost. Usporedba konstatnog i Gouraudovog sječanja pak pokazuje razliku koju donosi malo složeniji račun na prikaz osvjetljenja tijela. Iako se u programu učestalo računaju transformacije pogleda i perspektivna projekcija, one se računaju pomoću OpenGL-ovih funkcija koje su ipak puno brže nego funkcije napravljene u prethodnom zadatku.

4.2 Zadatak 9. - Fraktali

4.2.1 Kratak opis programa

Potrebno je nacrtati Mandelbrotov i Julijev skup za točke ograničene kompleksne ravnine koje su transformirane u odgovarajuće točke prozora na kojem iscrtavamo zadane skupove. Osnovni način iscrtavanje tih skupova je da jednom bojom označimo točke za koje niz konvergira, a drugom bojom niz koji divergira.

4.2.2 Promjene načinjene s obzirom na upute

U ovom zadatku nisu načinjene nikakve posebne promjene u uputama. U uputama piše da za parametar k koji predstavlja broj iteracija izračunamo određenu boju. Taj izračun može trivijalan ako koristimo samo dvije boje ili malo složeniji. Za određivanje boje korištena je funkcija iz službene skripte predmeta Interaktivna računalna grafika.

4.2.3 Korištene strukutre podataka

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

4.2.4 Upute za korištenje programa

Sam program je napisan u jednoj .cpp datoteci bez korištenja ikakvih drugih vlastitih biblioteke izuzev biblioteke freeglut potrebne za rad u OpenGl-u. Stoga je dovoljno program smjestiti u isti direktorij u kojem se nalazi biblioteka freeglut i zatim upisati naredbu za prevođenje programa g++ -Iinclude -Llib -o program.exe program.cpp -lfreeglut -lopengl32 -lglu32. Izvođenjem te naredbe stvara se izvršna datoteka program.exe koju možemo pokrenuti s parametrima u naredbenom retku ukoliko su oni potrebni.U ovom slučaju nemamo nikakve argumente u naredbenom retku. Nakon pokretanja programa potrebno je unijeti parametre koje su potrebne za izračun skupova,a to su prag epsilon, broj iteracija, granice kompleksne ravnine i kompleksni broj c. Pritiskom tipke p možemo prebacivati sliku na ekranu to jest mijenjati prikaz između dvije vrste skupova.

4.2.5 Komentar rezultata

Program iscrtava fraktale te se za određivanje konvergencije koristi određen broj iteracije.Taj postupak određivanja može biti poprilično zahtjevan te za stotinjak iteracija po jednoj točki taj postupak može trajati duže to jest slika se ne iscrtava trenutno. Za bojanje koristimo jednostavnu funkciju koja za točke koje konvergiraju pridružuje crnu boju,a za točke koje divergiraju pridružujemo boju u ovisnosti o tome u kojoj smo iteraciji odredili da točka ne pripada skupu.