SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
INTERAKTIVNA RAČUNALNA GRAFIKA
Ak. god. 2019./2020.

DOKUMENTACIJA

Laboratorijske vježbe - inačica A

Studentica: *Jelena Bratulić*
JMBAG : 0036506909

Zagreb, svibanj 2020.

Sadržaj

1. Osnovne informacije

Rješenja laboratorijskih vježbi većinski su implementirana u programskom jeziku Python. Iznimka je druga vježba iz prvog ciklusa vježbi koja je implementirana u programskom jeziku C/C++.

Svrha laboratorijskih vježbi je primijeniti znanje stečeno na predavanjima te samostalnim učenjem iz ostalih izvora. Većina laboratorijskih vježbi je implementirana po naputku iz službene knjige kolegija Interaktivna računalna grafika (`http://www.zemris.fer.hr/predmeti/irg/knjiga.pdf`).

Također, svaka vježba je implementirana prema uputama za pojedini ciklus i vježbe koje su dostupne na službenim stranicama kolegija(`http://www.zemris.fer.hr/predmeti/irg/laboratorijske_vjezbe.html`).

2. Prvi ciklus laboratorijskih vježbi

Prvi ciklus laboratorijskih vježbi implementiran je dijelom koristeći progamski jezik Python i razvojno okruženje PyCharm te dijelom koristeći programski jezik C i razvojno okruženje CodeBlocks. Prvi ciklus upoznao nas je s osnovnim matematičkim i vektorskim izrazima koji se koriste u interaktivnoj računalnoj grafici. Također, podsjetili smo se osnova linerane algebre koje će nam biti potrebne u daljnjem tijeku laboratorijskih vježbi. I na poslijetku, bavili smo se implementacijom Bresenhamovog algoritma za crtanje linija u rasterskom sustavu.

2.1 Vježba 1

Vježba 1 bila je podijeljena u tri manja podzadatka. U prvom podzadataku trebalo je implementirati osnovne operacije za računanje u lineranoj algebri. Budući da sam vježbu 1 implementirala u programskom jeziku Python, na raspolaganju mi je bila i biblioteka NumPy koja podržava operacije linerane algebre te rad s matricama. Budući da je zadatkom bilo zadano da se implementiraju operacije zbrajanja, oduzimanja, množenja vektora te vektorskog i skalarnog produkta, kao i rješavanja sustava matrica, zbog samog korištenja biblioteke numpy, nije bilo potrebno implementirati metode koje računaju spomenute operacije. Operacije koje sam koristila pri radu s matricama su:

- np.array() - za stvaranje matrice
- np.add(v1, v2) - za zbrajanje vektora v1 i v2
- np.vdot(v1, v2) - za računanje skalaranog produkta vektora v1 i v2
- np.cross(v1, v2) - za računanje vektorskog produkta vektora v1 i v2
- np.linalg.norm(v1) - za računanje norme vektora v1
- np.matmul(mat1, mat2) - za množenje matrica mat1 i mat2
- np.linalg.inv(mat1) - za računanje inverza matrice mat1

U drugom podzadatku koristeći operacije iz prvog podzadatka implementirali smo rješavanje lineranog sustava koji je zapisan matrično. Korisnik je unio podatke o 3 sustava s 3 nepoznanice (x, y, z), a zatim je program koji je implementiran, ukoliko je to moguće, izračunao rješenja sustava, odnosno vrijednosti za nepoznanice x, y i z.

Budući da je sustav ostvaren preko dvije matrice pri čemu su u matrici A koeficijenti uz nepoznanice, a u matrici B vrijednosti sustava, rješenje sustava određeno je kao x = A^-1*B. Naravno, rješenje je moguće dobiti samo ukoliko matrica A nije singularna, dakle ukoliko njezina determinantna nije jednaka nuli. Operacije koje sam koristila u ovom podzadatku su np.solve(mat1, mat2) za izračun rješenja x te np.linalg.det(mat1) za provjeru vrijednosti determinante te operacije koje su spomenute u opisu prvog podzadatka.

U trećem podzadatku smo se bavili pojmom baricentričnih koordinata. Korisnik je unio podatke o vrhovima trokuta (A, B, C) te o točki koju želi ispitati, a zatim je program izračunao i ispisao vrijednosti baricentričnih koordinata. Baricentrične koordinate omogućuju određivanje odnosa točke naspram vrhova trokuta, odnosno njihove vrijednosti nam ukazuju na to gdje se nalazi točka s obzirom na trokut.

Ukoliko je točka unutar trokuta, tada vrijedi da je t1 + t2 + t3 = 1, dakle zbroj svih baricentričnih koordinata je 1. Ukoliko je zadana točka u vrhu trokuta, tada će baricentrična koordinata koja odgovara tom vrhu biti 1, a ukoliko se točka nalazi u težištu trokuta, baricentrične koordinate će biti istog iznosa i iznosit će 1/3.

Određivanje baricentričnih koordinata implementirano je koristeći drugi podzadatak, odnosno kao rješavanje sustava jednadžbi. U matrici A zapisane su vrijednosti vrhova trokuta, dok su u matrici B vrijednosti koordinata ispitne točke.

Problem singularnosti matrice A pojavljuje se u dva slučaja. Ukoliko je neki vrh trokuta u ishodištu, matrice se transformiraju na način da se podatak o tom vrhu izbaci te se sustav pretvara u dimenzionalnost 2x2. Ukoliko su sve točke zadane u istoj ravnini, npr z=0 kod svih točaka, tada se taj koristi uvijet da je t1+t2+t3 = 1 te se nule u matrici zamjenjuju jedinicama.

2.2 Vježba 2

U drugoj vježbi zadatak je bio implementirati Bresenhamov algoritam za crtanje linija u rasterskim sustavima. Ova vježba je implementirana u programskom jeziku C i koristeći dobiveni predložak koda. U predlošku koda dobiven je osnovni algoritam koji radi samo za kuteve od 0 do 45 stupnjeva.

Problem kod osnovnog algoritma je što ne funkcionira za kuteve izvan spomenutog raspona, primjerice ukoliko bismo nacrtali liniju pod kutom od 60 stupnjeva, osnovni algoritam bi svejedno iscrtao liniju pod kutom od 45 stupnjeva jer implementacijski gledano, nemoguće je ostvariti pomicanje po y osi za više od 1, a nama je potrebno pomicanje za više od 1. Nadalje, osnovni algoritam je ograničen uvjetima položaja točke koji nama za potpunu funkcionalnost algoritma ne odgovaraju.

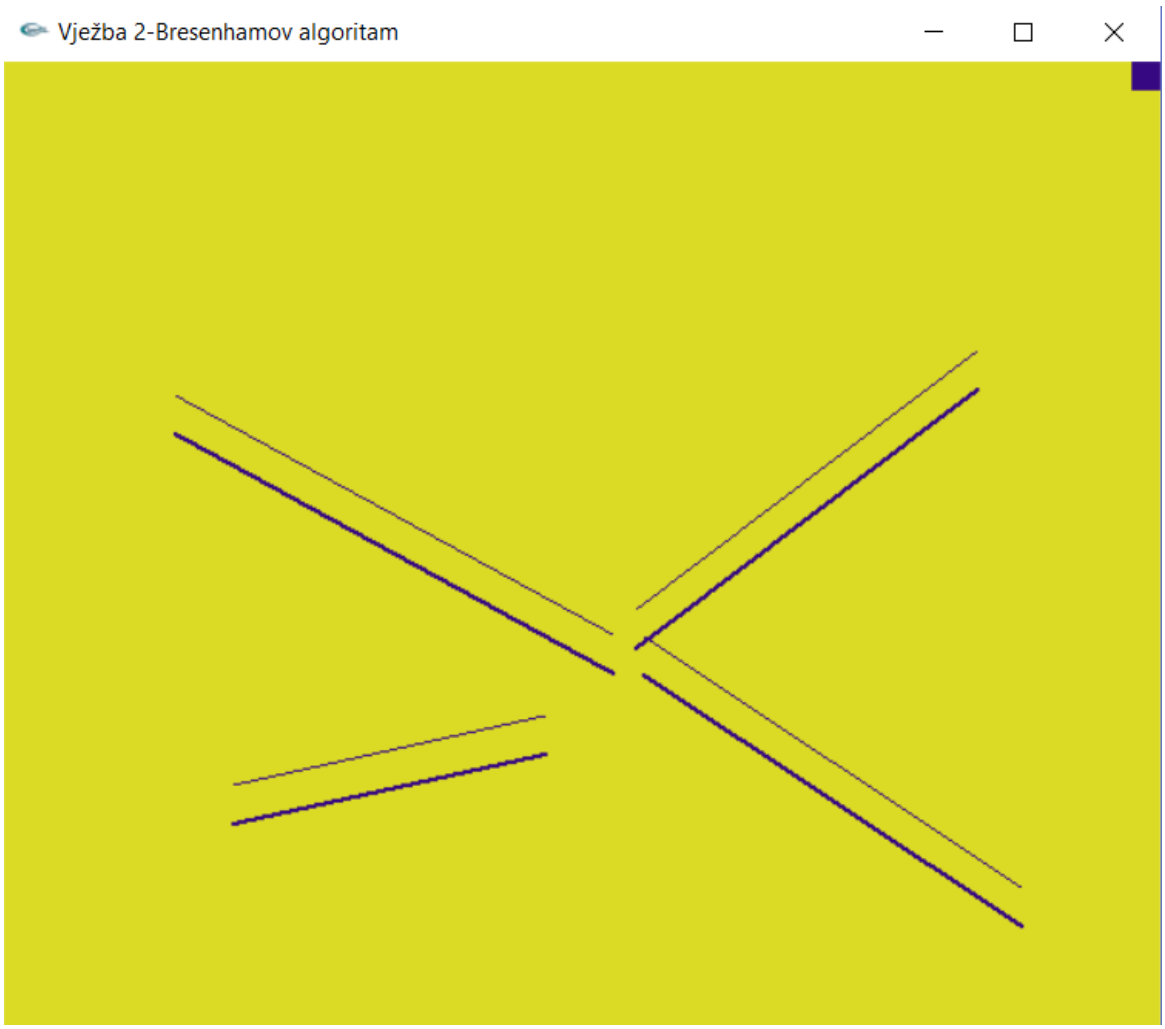
Kako bismo ostvarili funkcionalan algoritam, bilo je potrebno uvesti nekoliko promjena. Implementirala sam dvije verzije algoritma, jedan s decimalnim i jedan s cijelim brojevima, no budući da je algoritam s cijelim brojevima bolji, opisivat ću promjene na njemu. Algoritam za cijele brojeve je implementiran po naputku iz knjige čiji je link dan u prvom poglavlju ove dokumentacije.

Kako bismo ostvarili funkcionalnost od 0 do 90 stupnjeva, jednostavno smo zamijenili x i y koordinate ukoliko je tangest kuta veći od 1, dakle ukoliko je kut veći od 45 stupnjeva. Nadalje, kako bismo ostvarili funkcionalnost od 0 do -90 stupnjeva, bilo je potrebno, uz provjeru tangesa, dodati smanjenje po osi y, a ne povećanje, jer se krećemo u suprotnom smjeru.

I konačno kako bismo ova dva novoizvedena slučaja proširili na raspon od 0 do 360 stupnjeva, morali smo još odlučiti kada je potrebno pozvati koju funkciju i s kojim redoslijedom točaka. Budući da smo ostvarili funkcionalnost po područjima od 90 stupnjeva, imat ćemo 4 slučaja, a za pojedino ćemo se odučiti ovisno o položaju početne i krajnje točke.

Ukoliko je krajnja točka desnije od početne točke (gledano po x koordinati), crtamo normalno implementiran algoritam koji bi i trebao crtati s lijeva na desno. Ukoliko to nije zadovoljeno, bit će potrebno zamijeniti redoslijed početne i krajnje točke. Nadalje, potrebno je još provjeriti i odnose y koordinata točaka čime ćemo odrediti trebamo li iscrtavati liniju po kriteriju za pozitivne (0-90 stupnjeva) ili negativne(0-(-90) stupnjeva) kuteve. Zapravo su svi kriteriji određivanja za pojedini kut izvedeni iz definicije tangensa kuta.

Korisniku se prilikom pokretanja programa ispisuju upute o korištenju programa. Točke za iscrtavanje linije se zadaju pritiskom miša. Dodatne opcije koje su implementirane poput animacije izmjene boje pozadine i linije se aktiviraju ili isključuju pritiskom tipke koja je navedena u uputi. Dodatno, prilikom zadavanja nove točke, korisniku se prikaže informacije o kooridnatama točke. Korisnik završava izvođenje programa pritiskom na znak X u gornjem desnom rubu prozora. Rezultat izvođenja je prikazan na slici **??**.



Slika 2.1: Simulacija Bresnehamova algoritma za crtanje linije uz usporedbu s već implementiranom operacijom OpenGL-a za crtanje linija.

3. Drugi ciklus laboratorijskih vježbi

Druga laboratorijska vježba u cjelosti je implementirana koristeći programski jezik Python i razvojno okruženje PyCharm. Sama vježba sastojala se od dva zadatka, odnosno od vježbe 3 u kojoj se crtao poligon te su se ispitivale njegove karakteristike. U vježbi 4, crtalo se trodimenzionalno tijelo koje se sastoji od više poligona, ali se ono prikazivalo kao dvodimenzionalno u z=0 ravnini, dakle xy ravnini.

3.1 Vježba 3

Zadatak ove vježbe bio je iscrtati poligon, obojati ga te provjeriti položaj proizvoljne točke za taj polinom. Kako bi se implementirao ovaj zadatak, korištene su dodatne biblioteke NumPy i PyGlet za programski jezik Python.

Zadatk je implementiran na način da se prvo učitaju koordinate vrhova i ispitnih točaka iz datoteke. Zatim se ispišu podaci o broju učitanih vrhova i točaka te njihovih koordinata. Nakon toga na zaslonu se pojavljuje skočni prozor s iscrtanim poligonom koji je učitan te su također označene točke, ukoliko su one pročitane iz datoteke.

Korisniku se ispiše uputa za korištenje, odnosno da se lijevim klikom miša dodaju točke za ispitivanje koje se odmah iscrtavaju na zaslon te se za njih daje povratna informacije jesu li unutar ili izvan poligona. Desnim klikom miša korisnik pokreće bojanje poligona ukoliko poligon nije obojan, ili miče boju s poligon ukoliko je poligon je već obojan. Korisnik završava rad programa pritiskom na gumb X u gornjem desnom rubu.

Sam kod organiziran je u klasu Polygon u kojoj se nalaze svi podaci i metode za rad s poligonom te za ispitivanje njegovih karakteristika. Klasa se sastoji od nekoliko rječnika u kojima se pamte podaci o kooridnatama vrhova, ispitinoj točci ukoliko je pročitana iz datoteke te koeficijenti bridova. Također, implementirane su metode za računajne koeficijenata bridova, ispitivanje položaja točke te bojanje poligona.

Za ispitivanje položaja točke koristila se provjera vektorskog umnoška točke

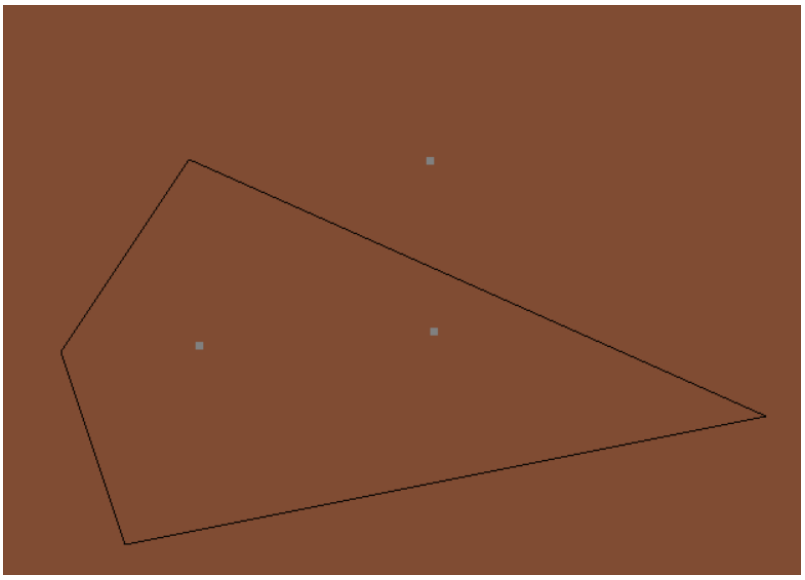
s bridovima. Dakle, ukoliko je umnožak točke s barem jednim bridom veća od nule, znamo da je točka izvan poligona i ne moramo ispitivati za ostale bridove, a ukoliko je za sve bridove taj umnožak manji od nule, onda smo sigurni da je točka unutar poligona.

Nadalje, prilikom implementiranja opcije za bojanje poligona, korišten je naputak iz upute za laboratorijsku vježbu. Sama logika iza tog algoritma je prilično jednostavna. Iterira se po svim y-vrijednostima od najmanje do najveće i provjerava se na kojoj x koordinati poligon siječe pravac $y = y_{trenutni}$. Nakon toga se crta se linija po tim koordinatama. Samo crtanje nije ostvareno odmah u funkciji već se u varijablu klase self.crtanje spremaju podaci o početnoj i krajnoj točki dužine koja će se iscrtati. Budući da je kao biblioteka za baratanje s grafikama korišten pyglet, svo potrebno crtanje ostvareno je u dekoratoru on_draw.

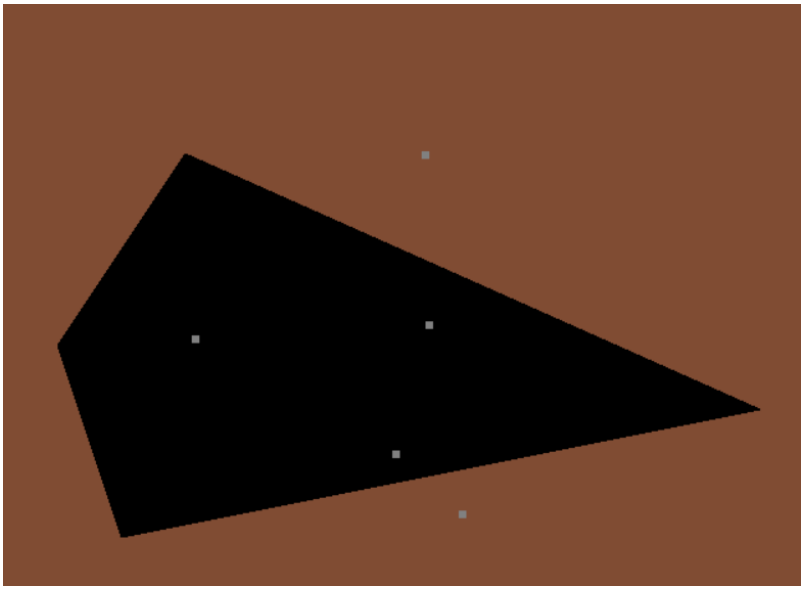
Konačno, na slikama je prikazan i izgled skočnog prozora u kojem se iscrtava i boji poligon te pomoću kojeg je moguće interaktivno dodati točke za ispitivanje. Također je prikazan i ispis u komandnoj liniji, može se primjetiti kako se prati unos točaka te akcije miša za bojanje poligona.

```
Ovo je program za crtanje konveksnog poligona i provjeru položaja točke.
Vrhovi poligona čitaju se iz datoteke.
Točke za provjeru moguće je pročitati iz datoteke ili interaktivno dodati klikanjem na zaslon.
--------------------------------------------------------------------------------
Upute:
Lijevi klik miša - dodavanje točke
Desni klik miša - promjena boje  poligona
--------------------------------------------------------------------------------
Broj vrhova poligona: 4
Koordinate 1. vrha: 50.0, 200.0
Koordinate 2. vrha: 150.0, 350.0
Koordinate 3. vrha: 600.0, 150.0
Koordinate 4. vrha: 100.0, 50.0
Kliknite na zaslon kako biste dodali točke za provjeru.
```

Slika 3.1: Pokretanje programa uz prikaz ispisa o položaju točaka.



Slika 3.2: Poligon s označenim točkama koje se ispituju prije bojanja. Bojanje se pokreće desnim klikom miša.



Slika 3.3: Obojani poligon s vidljivim oznakama točaka koje se ispituju.

3.2 Vježba 4

U ovoj vježbi zadaci su bili vezani uz trodimenzionalna tijela koja su izgrađena od poligona. Zadatak je bio da se učitaju podaci o tijelu, obrade, izračunaju potrebni dodatni podaci i koeficijenti, iscrta tijelo u z=0 ravnini te da se, ukoliko je tijelo konveksno, provjeri položaj točke u ovisnosti o tijelu.

Problem je implementiran u programskom jeziku Python uz korištenje dodatnih biblioteka Numpy i Pyglet, a sam kod je strukturiran u klasu Tijelo koje sadrži sve potrebne podatke i metode za baratanje s 3D tijelima. Unutar klase implementirane su metode za traženje rubnih koordinata, dakle ekstrema, računanje koeficijenata pojedine ravnine koja je određena vrhovima poligona, metoda za provjeru položaja točke te metode za translaciju i skaliranje tijela.

Program prilikom pokretanja ispiše popis mogućih tijela za crtanje i provjeru te se korisniku nudi mogućnost odabira jednog od ponuđeneih. Nakon uspješnog odabira tijela, ispišu se podaci o tijelu te ukoliko je tijelo konveksno, od korisnika se traži da unese koordinate točke V za koju želi ispitati odnos točke i poligona.

Prilikom implementacije zadatka, nad točkama tijela je provedno nekoliko transformacija. Prvo su određene rubne koordinate te je izračunato središte tijela kao aritmetička sredina rubnih koordinata. Zatim je tijelo pomaknuto za negativni iznos središta tijela kako bi središte tijela upalo u ishodište koordinatnog sustava. Nadalje, tijelo je skalirano na raspon [-1, 1] te su točke sada spremne za daljnju obradu. Nakon svih transformacija izračunati su koeficijenti pojedinih ravinina koje su opisane s tri točke tijela. Koeficijenti su izračunati po naputku iz upute za laboratorijsku vježbu. Dakle iznos determinante točaka koje određuju tu ravninu.

Koeficijenti ravnina su potrebni kako bismo za konveksna tijela mogli ispitati položaj točke u odnosu na tijelo. Budući da je na predavanju definirano kako skalarni produkt točke i ravnine daje informaciju o njihovom međusobnom odnosu, na taj isti način je definirana provjera položaja i u mojoj implementaciji. Naravno, ovaj postupak po kojem sam računala vrijedi ukoliko su svi vrhovi tijela zadani u smjeru kazaljke na satu gledano iz vana te ukoliko su normale ravnine usmjerene izvan tijela. Točka će biti izvan tijela ukoliko je barem jedan skalarni produkt točke i ravnine veći od nule. Ukoliko su svi skalarni produkti manji od nule, tada znamo da je točka unutar tijela.

Provjeru položaja točke za konkavna tijela nisam implementirala budući da se nije tražilo, ali kod konkavnih spomenuti postupak ne vrijedi jer ne mora značiti

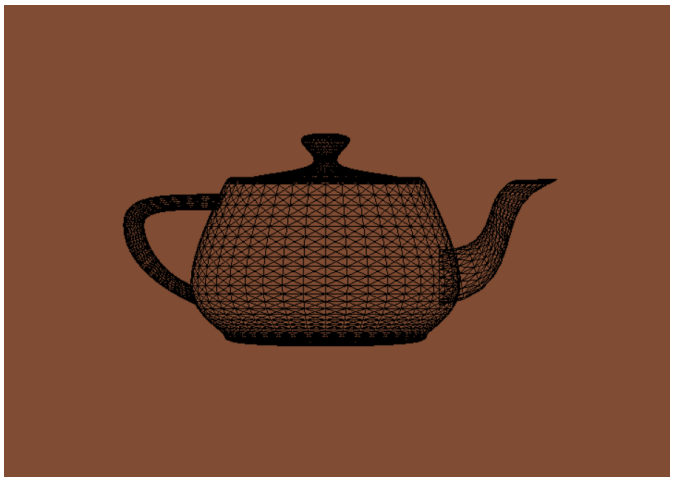
da će uvijek biti zadovoljen uvijet da normala ravnine gleda izvan tijela. Kod konkavnih se provjera radi na način da se gleda parnost probodišta točke s ravninom.

Na sljedećim slikama su prikazani rezultati pokretanja programa. Tijela su crtana u dvodimenzionalnom prostoru uz postavljenu z=0.

```
Mogući odabiri 3D tijela >> {1: 'all.obj', 2: 'bird.obj', 3: 'bull.obj', 4: 'dragon.obj', 5: 'frog.obj', 6: 'kocka.obj', 7: 'porsche.obj', 8: 'skull.obj', 9: 'teapot.obj', 10: 'teddy.obj', 11:
 'temple.obj', 12: 'tetrahedron.obj'}
Molimo odaberite jedan broj iz prethodnog izbornika >> 6
Ime: kocka| Vrhovi: 8| Bridovi: 12
Upišite x, y, z koordinatu točke V odvojene zarezima >> 2,3,4
Provjeravam položaj točke V(2.0, 3.0, 4.0)
Izvan je!
```

Slika 3.4: Pokretanje programa uz prikaz ispisa o položaju točke.

Slika 3.5: Iscrtavanje čajnika pomoću funkcije GL_POLYGON



Slika 3.6: Žičano iscrtavanje čajnika pomoću funkcije GL_LINES

4. Treći ciklus laboratorijskih vježbi

U trećoj laboratorijskoj vježbi bavili smo se problemom perspektiva i transformacija u prostoru te Bezierove krivulje. Obje vježbe su implementirane u programskom jeziku Python uz korištenje grafičke biblioteke Pyglet.

4.1 Vježba 5

U ovoj vježbi zadatak je uključivao transformaciju pogleda te prostornu perspektivu. Zadatak je bio da se nadogradi vježba 4 na način da se tijela prikažu u prostornoj perspektivi. Prostorna perspektiva je drukčija u odnosu na ortogonalnu ili vertikalnu perspektivu te ju najjednostavnije možemo shvatiti kao da gledamo na scenu kroz malu rupicu, odnosno kroz mali prozorčić.

Kod same vježbe organiziran je kao i u vježbi 4, dakle u klasi Tijelo koja sadrži sve potrebne metode za manipulaciju nad objektom. Razred Tijelo ima i varijable razreda u kojima su pohranjeni podaci o ekstremima tijela te o točkama i koeficijentima ravnina. Rječnik u kojem su pohranjene točke i oznake ravnine je oblike redni broj točke —¿ (x, y, z, h) točke, analgono tome za ravnine vrijedi: redni broj ravnine —¿ t1, t2, t3 koje označavaju poligon te ravnine.

U ovoj vježbi uvodimo i još dva nova pojma - očište i gledište. Očište je točka kroz koju gledamo, primjerice naše oko, dok je gledište točka u koju gledamo. Gledište je najčešće stavljeno u središte tijela, na taj način dobivamo centriranu sliku prilikom prikaza tijela. Prilikom zadavanja očišta moramo paziti na jednu stvar, a to je da očište mora biti zadano izvan raspona tijela. Dakle u našem slučaju, budući da će tijelo biti normalizirano na raspon [-1, 1], moramo zadati točke koje su izvan tog raspona.

Budući da sam uzela kod iz prošle vježbe 4, tijela su već bila učitana i normalizirana na raspon [-1,1]. No, bilo je potrebno provesti transformaciju pogleda na način da se izračunaju vrijednosti matrica translacije i rotacije. Matrice su implementirane po naputku iz upute te uz korištenje standardnih operacija za rad s matricama iz biblioteke numpy.

Finalna matrica transformacije pogleda je zapravo umnožak svih transformacija

koje je potrebno provesti:

- translacija ishodišta u točku očišta
- rotacija oko z osi
- rotacija oko y osi
- rotacija za 90 stupnjeva oko z osi
- promjena predznaka na x osi

Prilikom svake transformacije, potrebno je tranfsormirati i točku gledišta budući da se cijeli prostor transformira. Konačno, matrica prostorne perspektive je finalna matrica kojom se zaključuje transformacija u prostoru.

Nakon provedene transformacije, preostaje još iscrtati tijelo. Tijela su iscrtana u žičanoj formi uz korištenje funkcije GL_LINES. Dodatno je još implementirana mogućnost pomicanja točaka gledišta i očišta pritiskom na odgvarajuće tipke. Kako bi se korisnik lakše snašao, prilikom pokretanja programa, ispiše se uputa u kojoj je opisano kojom tipkom je moguće promijeniti točke. Finalni rezultati prikazan je na sljedećoj slici.

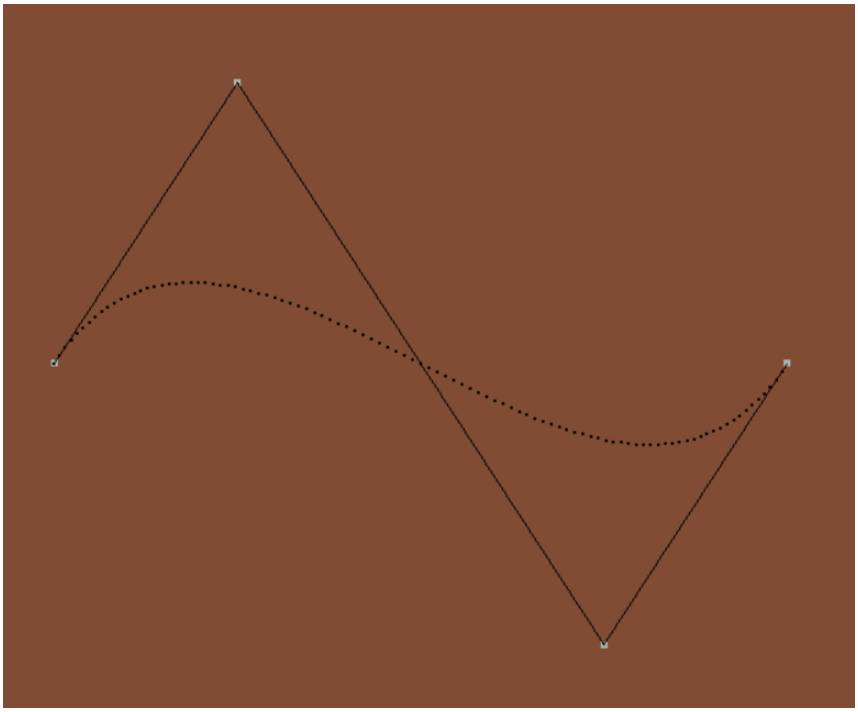
Slika 4.1: Iscrtavanje čajnika prilikom pokretanja programa. Sam prikaz tijela uvelike ovisi o odabiru točke očišta.

4.2 Vježba 6

U ovoj vježbi bavili smo se aproksimacijom Bezierove krivulje te kretanjem tijela po Bezierovoj krivulji uz sakrivanje stražnjih poligona. Prilikom implementacije zadatka, koristila sam kod iz vježbe 5 budući da ću u finalnom koraku ostvariti kretanje tijela po Bezierovoj krivulji. Struktura koda ove vježbe opisana je u Vježbi 5 i potpuno je identična.

Bezierovu krivulju možemo definirati na dva načina. Prvi način uključuje Bezierove težinske funkcije, a drugi polinome Bersteina. U implementaciji zadatka koristila sam drugi način, dakle aproksimaciju krivulje preko polinoma Bersteina budući da sam koristila kontrolne točke. Aproksimirana krivulja poslužila mi je kao putanja za kretanje tijela. Sam princip rada Bersteinovih polinoma je jednostavan i uključuje računanje koeficijenata preko sljedeće formule: $\frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}$.

Ideja je da se aproksimira krivulja uz jako mali korak pomicanja (t). Ja sam koristila povećanje od 0.01 u granicama [0,1]. Prilikom aproksimacije Bezierove krivulje, važno je napomenuti kako će aproksimacija prolaziti kroz početnu i krajnju točku, ali ne i kroz ostale, prema njima će se samo kretati. Primjerak stvorene Bezierove krivulje prikazan je na sljedećoj slici.

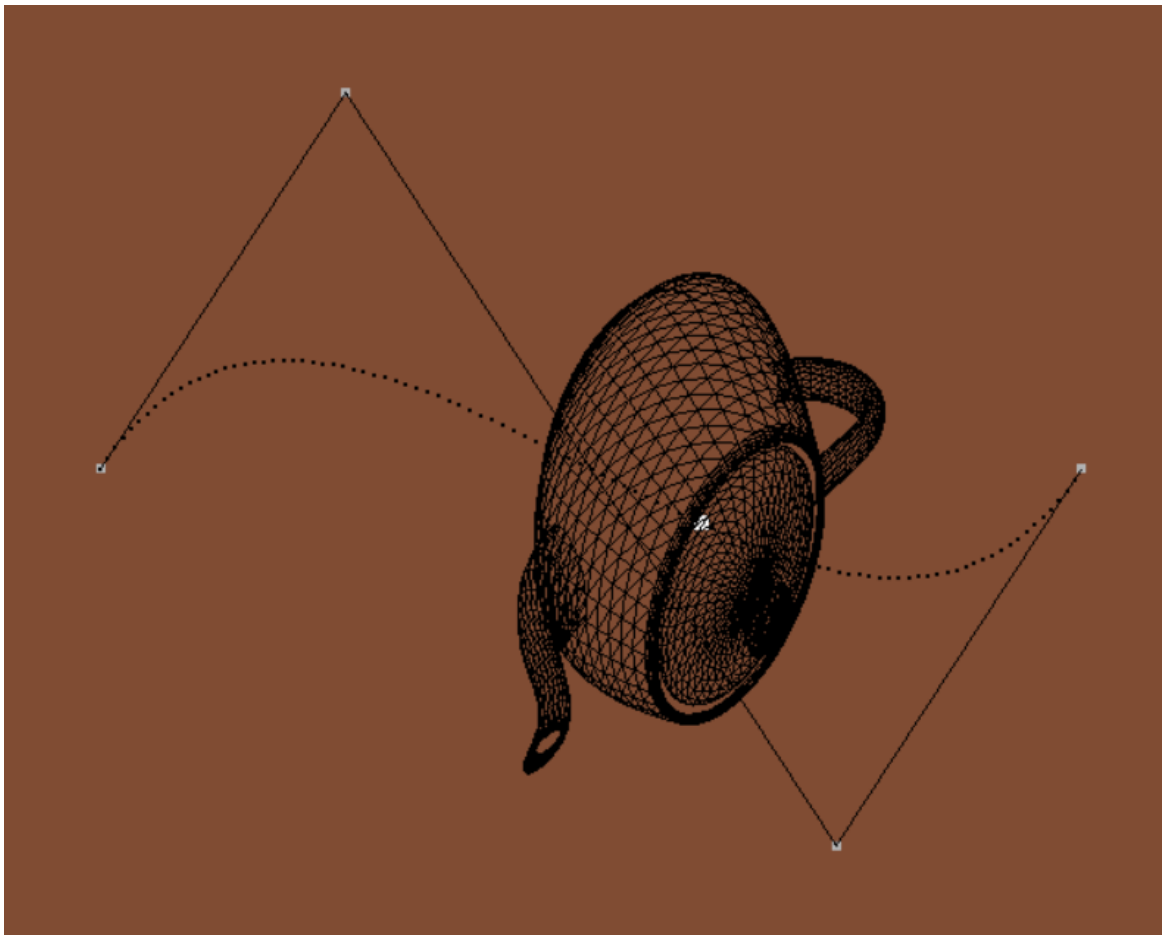


Slika 4.2: Aproksimirana Bezierova krivulja u odnosu na 4 kontrolne točke.

Sljedeći korak je bio da se tijelo giba po stvorenoj krivulji. No, kako bi se tijelo

lakše i brže kretalo po krivulji, bilo je potrebno maknuti stražnje poligone. Micanje je napravljeno na način da se ispitivao kut koji zatvaraju ravnina poligona i vektor iz točke očišta. Ukoliko je kosinus kuta veći od nule, poligon je prednji i iscrtava se, no ukoliko je manji od nule, poligon je stražnji i ne iscrtava se.Radi jednostavnosti, ja sam pomicala središte tijela na način da se u svakom koraku aproksimacije, sve točke tijela pomaknu u točku aproksimacije.

Implementirala sam animaciju uz korak pomicanja od 0.1 sekunde. Korsniku se prilikom pokretanja programa ispisuje uputa o korištenju programa. Korisnik može pritiskom na tipku A, zaustaviti ili pokrenuti kretanje tijela po krivulji, a također je potrebno i da se zada ime datoteke iz koje se čitaju kontrolne točke.



Slika 4.3: Kretanje tijela po Bezierovoj krivulji. Bijelom točkom je označeno središta tijela te se moguće pratiti kako se ono podudara s aproksimacijom.

Dodatno je, u zadnji tren zbog naknadne izmijene zadatka, dodana i verzija programa u kojoj se očište pomiče po aproksimizanoj Bezierovoj krivulji. U toj verziji programa se korisniku pričinja kao da se kreće oko predmeta kojeg proma-

tra. Budući da je krajnji rezultat tog programa kratki video kretanja oko tijela, a video se ne može priložiti u dokumentaciji, prikaz rezultata je izostavljen iz ove dokumentacije.

5. Četvrti laboratorijskih vježbi

U četvrtoj vježbi bavili smo se sjenčanjem tijela te fraktalima. Obje vježbe su implementirane u programskom jeziku Python i u razvojnoj okolini PyCharm. Također je korištena grafička biblioteka Pyglet te dodatne pomoćne biblioteke Numpy i Math.

5.1 Vježba 7

U ovoj vježbi zadatak je bila usporedba konstantog sjenčanja i Gourdaova sjenčanja. Iako je isti princip po kojem se računaju, odnosno oba za sjenačanje koriste ambijentalnu i difuzijsku komponentu, razlikuju se po načinu na koji se difuzijska komponenta računa. Ambijentalna komnponenta je definirana kao umnožak $I_i * k_a$ što je u mojoj implementaciji jednako $amb = 100 * 0.5 = 50$.

Bitna komponentna kod sjenčanja je i vektor izvora sjenčanja L. On je zadan razlikom točke koju promatramo i točkom izvora koju smo zadali, odnosno učitali. Vektor L doprinost će iznosu u difuznoj komponenti intenziteta.

Nadalje, kod konstantnog sjenčanja, cijela ravnina imat će istu boju, odnosno bit će s istim intentitetom osjenčana. Dakle, u ovom slučaju se difuzna računa kao $dif =$. Dakle, za svaku ravninu potrebno je odrediti koeficijente ravnine. Koeficijenti su izračunati na način koji je opisan u vježbi 4 iz drugog ciklusa laboratorijskih vježbi. Dobiveni intenzitet bit će u rasponu 0-1, to postižemo dijeljnjem s 255. Ukoliko je neki poligon stražnji, tada bi njegov intenzitet bio 0, odnosno ne bi bio osvjetljen. Kako bi se olakšalo crtanje osjenčanih poligona, stražnji poligoni se ne crtaju. Provjera stražnjih poligona napravljena je kako je opisano u vježbi 6 u trećem ciklusu laboratorijskih vježbi. Odnosno u ovom slučaju difuzna nije zastupljena uopce?

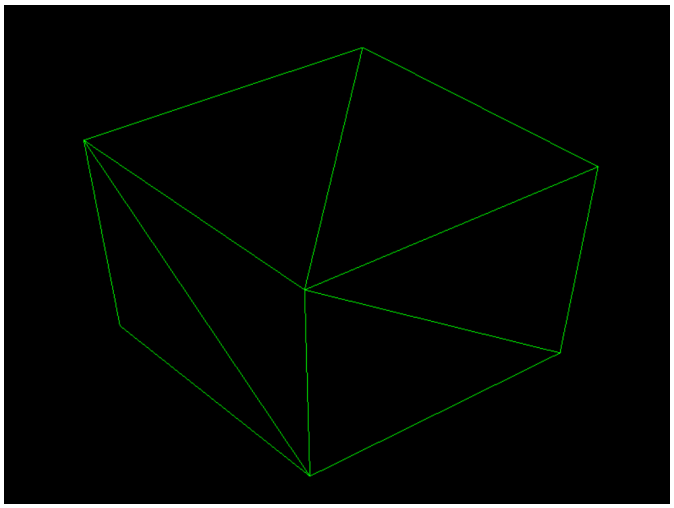
Kod Gourdavova sjenčanja je pojedini poligon osjenčan u više boja, odnosno intenziteti pojedinih vrhova koji određuju poligon se linerano interpoliraju na cijelu plohu ravnine. U ovom slučaju, difuzna komponenta se računa pomoću normiranih normala vrhova. Normirane normale vrhova su izračunate na način da se u pojedinom vrhu izračuna aritmetička sredina svih koeficijenata koji određeni tim

vrhovima te se taj vektor normale normira.

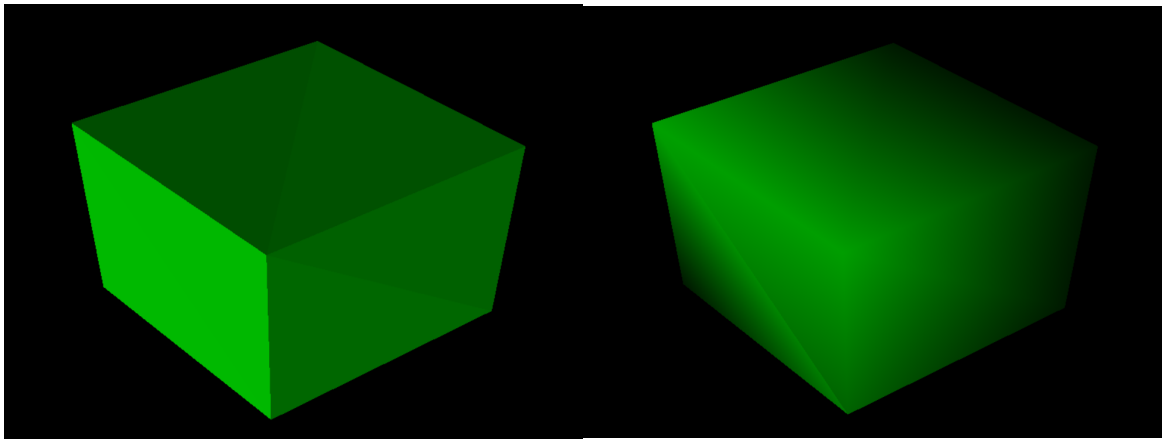
Program prilikom pokretanja ispisuje upute o korištenju. Naime, moguće je mijenjati koordinate točke očišta i gledšta kako bi se vidjele razlike u sjenčanju poligona. Potrebno je zadati i točku iz koje pada svijetlo za sjenčanje. Dodatnim tipkama omogućeno je mijenjanje vrste prikaza - žičani prikaz tijela, tijelo s konstantim sjenčanjem i tijelo s Gourdaovovim sjenčanjem.

Struktura koda preuzeta je iz Vježbe 5 i potpuno je ista uz dodatne izmijene pri pozivu funkcije za crtanje on draw. Unutar same funkcije on draw implementirano je crtanje ovisno o odabranom načinu radu, odonsno vrsti sjenčanja.

Rezultati pokretanja programa su prikazani na sljedećim slikama.



Slika 5.1: Žičani prikaz tijela kocke.



(a) Konstantno sjenčanje (b) Gourdaovo sjenčanje

Slika 5.2: Sjenčanje kocke u različitim načinima

5.2 Vježba 8

U ovoj vježbi bavili smo se računanjem i projiciranjem fraktala. Fraktali su geometrijski objekti čija je fraktalna dimenzija strogo veća od topološke dimenzije. U ovoj vježbi uvodimo pojam kompleksne ravnine te ravnine prikaza koju predstavlja naš zaslon prozora.

Kako bismo mogli projicirati Mandelbrotov ili Julijev fraktalni skup, bilo je potrebno prijeći u kompleksnu ravninu. Kompleksna ravnina određena je maksimalnim i minimalnim koordinatama $u_{min}, u_{max}, v_{min}, v_{max}$ koje učitavamo prilikom pokretanja programa, dok je ravnina prikaza određena maksimalnim vrijednostima prozora koji se otvara (u našem slučaju $x_{max} = 800, y_{max} = 600$).

Kako bismo se prebacili u kompleksnu ravninu, potrebno je izvesti sljedeću pretvorbu: $u_0 = \frac{u_{max} - u_{min}}{x_{max}} * x + u_{min},\ v_0 = \frac{v_{max} - v_{min}}{y_{max}} * y + v_{min}$. Zatim, kako bi izračunali elemente fraktalnih skupova, koristila sam formulu iz uputa za ovu vježbu. Ukratko, iteriramo po svim x i y po zaslonu ekrana i računamo pripadnu boju za tu poziciju. Broj iteracija određen je varijablom m, a prag varijablom ϵ. Određivanje boje vrši se na način da se povećava komponenta k dok god je ona manja od broja iteracija m te dok je r, koji predstavlja apsolutnu vrijednost vrijednosti funckije za taj k manja od praga, odnosno od ϵ.

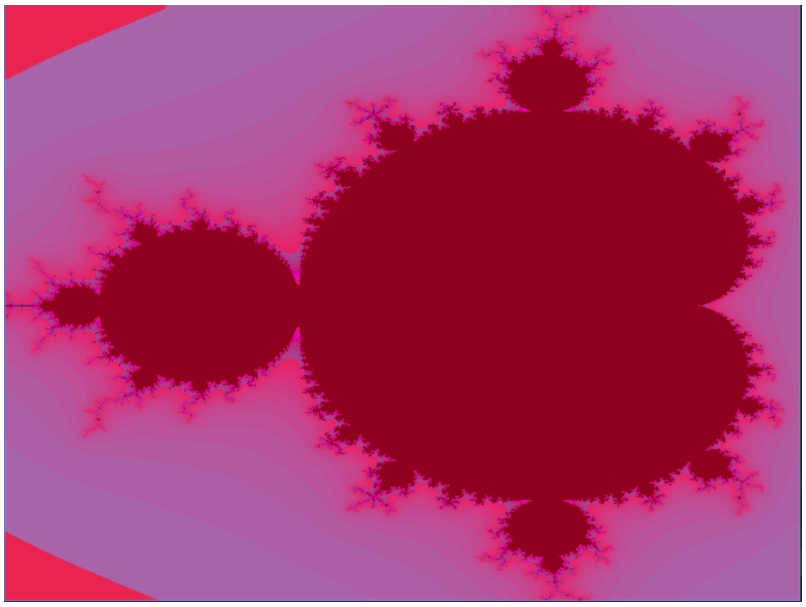
Strukutra koda ove vježbe je vrlo jednostavna. Nisam koristila nikakve dodatne razrede već je sva logika i rješavanje problem izračuna koeficijenata implenetirana unutar dekoratora on draw koji je potreban za crtanje na ekran ukoliko koristimo pyglet biblioteku.

Kako bismo prikazali raznovrsniji fraktalni skup, dodatno su korigirane boje te je stvoreno 5 kategorija u kojima se po drukčijim kriterijima računaju r, g i b komponente boje. Kategorije su podijeljene po odnosu k naspram m, a sve boje su namještene da budu u nijansama roze boje(rgb = 1.0, 0.0, 0.5).

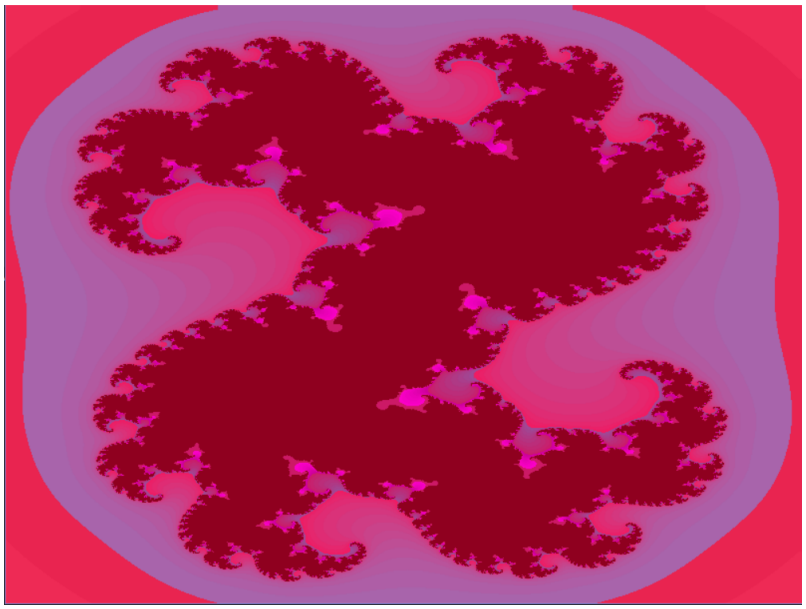
Razlika između Mandelbrotova i Julie skupa je samo u podešavanju početnih uvjeta. Kod Mandelbrotova skupa početni uvjeti su: $k = -1, c_{realno} = u_0(x), c_{imaginarno} = v_0(y), z_{realno} = z_{imaginarno} = 0$, dok kod Julie vrijedi: $k = -1, c_{realno} = c_r, c_{imaginarno} = c_i, z_{realno} = u_0(x), z_{imaginarno} = v_0(y)$.

Prilikom pokretanja programa, korisnik je dužan unijeti ime ulazne datoteke iz koje se čitaju potrebni podaci te poslati informaciju želi li prikazati Mandelbrotov ili Julie fraktalni skup. Rezultati pokretanja prikazani su na sljedećim slikama.

Budući da nisam implementirala višedretvenost kod izračunavanja i iscrtavanja



Slika 5.3: Prikaz Mandelbrotova fraktalnog skupa za parametre: $\epsilon = 100, m = 46, u_{min} = -1.5, u_{max} = 0.5, v_{min} = -1, v_{max} = 1$



Slika 5.4: Prikaz Julie fraktalnog skupa za parametre: $\epsilon = 100, m = 46, u_{min} = -1, u_{max} = 1, v_{min} = -1.2, v_{max} = 1.2, c_r = 0.32, c_i = 0.048$

fraktala, potrebno je pričekati 30 sekundi da se ekran iscrta. Ovo je problem koji je moguće doraditi.