SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET ZAGREB

ZAVRŠNI RAD

Ella Gracin

Sveučilište u Zagrebu
Grafički fakultet

Smjer: Dizajn grafičkih proizvoda

ZAVRŠNI RAD

PRIMJENA POSTSCRIPTA U PRIKAZU UMJETNIČKOG DJELA

Mentor:
doc.dr.sc. Tajana Koren Ivančević

Student:
Ella Gracin

Zagreb, 2016.

SAŽETAK

PostScript je grafički programski jezik pomoću kojega je moguće prikazati razne oblike koristeći jednostavne naredbe. Ideja ovog završnog rada je u njemu prikazati djelo nizozemskog slikara Vincenta van Gogha, čija se mentalna nestabilnost očitovala u slikama. Izazov će biti u dočaravanju njegovih energičnih poteza kista te pastoznih nanosa boje pravilnim vektorskim oblicima i linijama kakve je moguće iscrtati u PostScriptu. U ovom će radu biti pobliže opisani načini prikaza pojedinih dijelova Van Goghove slike „Zvjezdana noć“ u Postscriptu, a krajnji će rezultat pokazati koliko je detaljno moguće prikazati takvu sliku koristeći naredbe na računalu umjesto boje na platnu te je li moguće zadržati snažan dojam koji slika ostavlja na promatrača koristeći vektorsku grafiku.

Ključne riječi: PostScript, Vincent van Gogh

ABSTRACT

PostScript is graphic programming language that can make various shapes using simple commands. The idea of this work was to show an artwork of Dutch artist Vincent Van Gogh, whose mental instability was shown in paintings. It will be challenging to impress his energic brush strokes and pasty color using only vector shapes and lines that PostScript allows. In this final work ways of making each part of the“Starry night“ by Vincent Van Gogh in PostScript will be closely described and the final result will show how detailed it is possible to show a painting like that using commands on computer instead of colors on a canvas and is it possible to keep a strong impression that original painting leaves on a person using vector graphic.

Key words: PostScript, Vincent van Gogh

SADRŽAJ

1. UVOD

Razvoj računalne grafike i programa posljednjih je godina toliko napredovao da su danas čak i slikari u mogućnosti svoje umjetničke radove kreirati koristeći samo računala. U ovom radu jedna od slika nizozemskog slikara 19.stoljeća, Vincenta van Gogha, prikazana je pomoću PostScripta, koji koristi naredbe čijim se definiranjem iscrtavaju odgovarajući vektorski oblici. Izbor teme završnog rada omogućuje spajanje postimpresionističkog stila i ideja sa modernom izvedbom, a pokazuje koliko je vjeran prikaz slike moguće postići koristeći programski jezik primarno namijenjen vektorskom opisu stranice.

Iako Van Gogh u svoje vrijeme nije mogao niti zamisliti korištenje računala u svrhu stvaranja umjetnosti budući da su računala 19. stoljeća mogla izvršavati tek neke računske operacije i sortiranja, pitanje je da li bi se, kada bi živio u današnje vrijeme, ostavio kista i platna te okrenuo slikanju u digitalnom obliku. Naime, posebnost njegovog stila očituje se upravo u debelom nanosu boje na platnu koja se dijelom izgubi već i u fotografijama njegovih slika te se mora doživjeti, što bi se u digitalnom obliku potpuno izgubilo. Cilj ovog završnog je sliku „Zvjezdana noć“ prikazati što sličniju originalu. Već je na početku jasno da će cjelokupni dojam koji slika ostavlja biti teško dočarati na taj način uzevši u obzir da je nemirne i energične poteze kista potrebno prikazati pažljivim smještanjem odgovarajućih oblika u koordinatni sustav. Ipak, prenošenje ugođaja slike olakšavaju brojne krivulje koje je slikar koristio, kao i dinamična kompozicija slike. To su sve stvari koje znatno otežavaju izradu praktičnog dijela završnog rada, ali su zaslužne da slika i u digitalnom obliku u sebi zadrži nemir koji prevladava u originalnom djelu. U svakom slučaju može se očekivati kako će prikaz „Zvjezdane noći“ u PostScripu bez sumnje biti prepoznatljiv svakom poznavatelju umjetnosti i djela Vincenta van Gogha.

2. TEORIJSKI DIO

PostScript je programski jezik namijenjen opisu stranice, a predstavljen je kao prvi komercijalni proizvod tvrtke Adobe Systems. On u svojoj grafici koristi vektorske oblike, a staza po kojoj oni nastaju definirana je naredbama. Te su naredbe najčešće kratice ili skupovi engleskih riječi, a one mogu stajati same ili uz vrijednost koja određuje na koji način će se izvršavati. Takve vrijednosti stoje ispred naredbi, a ovisno o naredbi mogu označavati koordinate na kojoj se odvija, vrijednosti za debljinu linije, radijus, boju i slično. [1]

Naredbe se u PostScriptu mogu unositi putem tekst editora poput Notepada, a dovoljno ih je odvajati razmacima iako je zbog bolje preglednosti i lakšeg snalaženja u kodu poželjno svaku pisati u novom redu. Lakšem snalaženju pomaže i dodavanje komentara, što je se radi dodavanjem znaka % ispred teksta.

Ono što se želi prikazati upisujući naredbe može se vidjeti kao cijela stranica u preglednicima poput Gsviewa, iz čega se kasnije takva stranica može prebaciti u PDF format. Oblike prikazane na taj način moguće je prikazati na odvojenim stranicama, a prelazak na sljedeću stranicu omogućuje *showpage* naredba.

2.1. STACK

PostScript je zasnovan na *stack* procedurama, koje su osnovni princip rada. *Stack* funkcionira tako što se iz njega izvlače vrijednosti za pojedine naredbe na način da je vrijednost koja je zadnja unesena ona koja će biti prva iskorištena u naredbi. Nakon izvršenja naredbe, podatak se uklanja sa vrha stacka i na red za izvršenje dolazi prethodna vrijednost. Podatke je sa stacka moguće izbrisati upisivanjem naredbe *clear*.[1]

Stanje na stacku moguće je ispisati upisujući kao skup naredbi:

```
/s {mark pstack pop} def
```

2.2. KOORDINATNI SUSTAV

Koordinatni sustav pomoću kojega se na stranicu smještaju oblici počinje u donjem lijevom kutu stranice, a točke na koje je podijeljen dužine su 0.353 milimetara. Njegovu je veličinu i položaj ipak moguće promijeniti naredbama koje na njega utječu te tako

dobiti i željene promjene na oblicima koje se potom smještaju na stranicu. Naredbe koje utječu na oblik i položaj čitavog koordinatnog sustava su translate, rotate i scale. [2]

2.2.1. Translate

Translate je naredba koja se koristi za pomicanje ishodišta stranice i pomicanje koordinatnog sustava, a ispisuje se uz vrijednosti koje označavaju pomak po x i y osi. Tako bi se, u slučaju da je ishodište koordinatnog sustava potrebno pomaknuti za 100 točaka po x i y osi, u tekst editoru trebalo upisati „`100 100 translate`“.

2.2.2. Scale

Kao i *translate, scale* naredba uz sebe sadrži vrijednosti za x i y, ali u tom slučaju nije riječ o pomaku po koordinatnom sustavu. Brojevi koji stoje uz *scale* označavaju za koliko će se posto koordinatni sustav smanjiti ili povećati po osi x i y tako da broj 1 označava 100% vrijednosti koordinatnog sustava i ne mijenja njegov izgled. Naredba za smanjenje koordinatnog sustava i oblika koji se na njemu opisuju glasila bi „`0.5 0.5 scale`“ i u tom bi primjeru smanjila koordinatni sustav za 50%.

2.2.3.Rotate

Za razliku od *translate* i *scale,* naredba *rotate* u sebi sadrži samo jednu vrijednost, a ona označava kut za koji će se koordinatni sustav i sve na njemu opisano zakrenuti u smjeru kazaljke na satu. Za smjer suprotan smjeru kazaljke na satu broju se dodaje minus, a za rotaciju od 45 stupnjeva u smjeru suprotnom smjeru kazaljke na satu upisuje se naredba „`-45 rotate`“.

2.2.4. Gsave i grestore

Stanje koordinatnog sustava moguće je spremiti *gsave* naredbom kako bi se nakon promjena mogao vratiti naredbom *grestore*. Naredba *gsave* spremit će sve što je do tada bilo zadano, pa će se tako uz *grestore* vratiti i prethodno definirana boja, staza, oblik linije i ostalo.

2.3.DEFINIRANJE NAČINA PRIKAZA

2.3.1. Definiranje boje

Prije nego što se u PostScriptu definira i prikaže staza, potrebno je definirati način na koji će se prikazati. Ukoliko se prethodno ne definira boja obruba ili ispune, ona će se prikazati kao crna boja. To je moguće promijeniti definicijom boje preko RGB, HSB ili

CMYK sustava ili određivanjem postotka sive boje. Postotak sive boje definira se kao primjerice `1 setgray`, gdje 1 označava 100% i boja definirana na taj način prikazuje se kao bijela, dok se oblik ispunjen bojom definiranom kao `0 setgray` prikazuje u crnoj boji. [3] Slika 1 prikazuje tri kvadrata iste veličine, obruba definiranog kao 0 posto sive, a ispune redom 0.25 0.5 i 0.75 posto sive boje.

Slika 1. Prikaz različitih nijansi sive korištenjem naredbe *setgray*

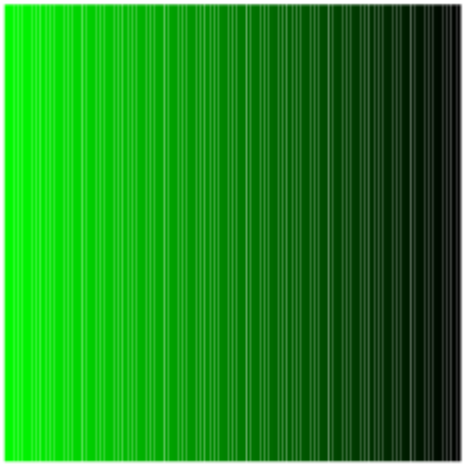
Program koji prikazuje takve kvadrate na stranici izgleda ovako:

```
%Prvi kvadrat
0.25 setgray
10 10 50 50 rectfill
0 setgray
10 10 50 50 rectstroke
%Drugi kvadrat
0.5 setgray
70 10 50 50 rectfill
0 setgray
70 10 50 50 rectstroke
%Treći kvadrat
0.75 setgray
130 10 50 50 rectfill
0 setgray
130 10 50 50 rectstroke
```

Vrijednosti za boju u RGB, HSB i CMYK sustavu definiraju se pojedinačno za svaku vrijednost u sustavu, a označavaju se brojevima od 0 do 1. Na taj se način boje definiraju po principu miješanja boja u svakom od tih sustava, a žuta se boja, primjerice, može opisati kao `1 1 0 setrgbcolor` ili `0 0 1 0 setcmykcolor`. U

PostSriptu je moguće postići dojam spontanog prijelaza boje uz pomoć promjene varijabli kroz petlju. Varijabla se izvan petlje definira kao željena početna vrijednost za boju, a istoj se unutar petlje dodaje ili oduzima vrijednost za onoliko koliko je potrebno kako bi u broju ponavljanja u petlji varijabla dosegla konačnu vrijednost. To se može pokazati u primjeru niza ravnih okomitih linija s pomakom od kojih svaka poprima vrijednost zelene boje u RGBu manji od prethodne te se tako kreću prema crnoj boji, kao što je prikazano na slici 2.

```
/g 1 def
100 {
/g g 0.01 sub def
0 g 0 setrgbcolor
0 0 moveto
0 100 lineto
stroke
1 0 translate} repeat
```



Slika 2. Prikaz spontanog prijelaza boje

Tako je vrijednost za boju prve linije definirana kao `0 1 0 setrgbcolor`, sljedeće `0 0.99 0 setrgbcolor`, a vrijednost za boju zadnje linije definirana je kao `0 0 0 setrgbcolor.`

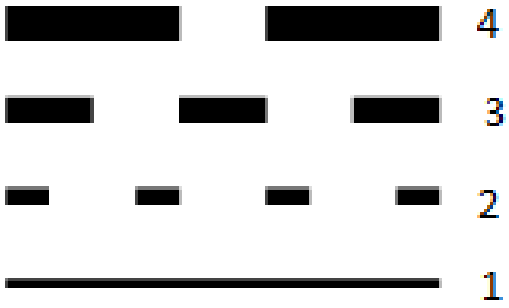
2.3.2. Definiranje oblika linije

Linije se također mogu opisati na brojne načine te im je moguće definirati debljinu,

isprekidanost, izgled ruba i način spajanja. Debljina linije definira se jednostavnom naredbom *setlinewidth* ispred koje se dodaje vrijednost za debljinu linije definirana u točkama. Hoće li linija biti konstantna ili isprekidana moguće je odrediti dodajući naredbu *setdash* uz koju se u četvrtastim zagradama upisuje odnos dužine linije i razmaka nakon kojega slijedi dužina nakon koje počinje niz izražena u točkama. Ako se isprekidanost linije ne unese kao naredba, linija će biti prikazana kao konstantna, što je moguće napisati i kao `[1 0] 0 setdash`. Vrijednosti unutar četvrtastih zagrada označavaju izmjenu punih i praznih dijelova unutar linije, a vrijednost nakon zagrade stoji za točku u kojoj će ta izmjena početi, odnosno može odrediti hoće li linija početi na ispunjenom ili praznom dijelu. Primjer dvije linije različite isprekidanosti i debljine u kodu :

```
%1
10 10 moveto
50 0 rlineto
1 setlinewidth
stroke
%2
10 20 moveto
50 0 rlineto
2 setlinewidth
[5 10] 0 setdash
stroke
%3
10 30 moveto
50 0 rlineto
3 setlinewidth
[10 10] 0 setdash
stroke
%4
10 40 moveto
50 0 rlineto
4 setlinewidth
[20 10] 0 setdash
stroke
```

Slika 3 prikazuje kako takav skup naredbi izgleda na stranici.

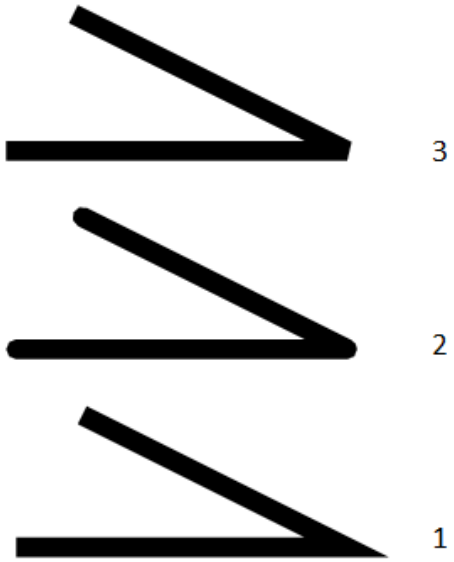


Slika 3. Linije različite debljine i isprekidanosti

Setlinecap i *setlinejoin* su naredbe kojima prethode vrijednosti od 0, 1 ili 2, a svaki od njih opisuje način na koji će se linija prikazati. U *setlinecap* naredbi vrijednost 0 označava kvadratni rub linije, kakav će se prikazati i u slučaju da rub linije ostane nedefiniran. `1 setlinecap` je naredba kojom bi se rub linije definirao zaobljen, a `2 setlinecap` bi definirala rub linije kvadratnog oblika produžen za pola debljine linije. Kod *setlinejoin* naredbe broj 0 označava spajanje dvije linije šiljastim vrhom, 1 bi označio dvije linije spojene zaobljenim vrhom, a `2 setlinejoin` bi definirao spajanje dviju linija tupim vrhom. [4] U primjeru koda to izgleda ovako:

```
3 setlinewidth
%1
10 10 moveto
50 0 rlineto
40 neg 20 rlineto
stroke
%2
10 40 moveto
50 0 rlineto
40 neg 20 rlineto
1 setlinejoin
1 setlinecap
stroke
%3
10 70 moveto
50 0 rlineto
40 neg 20 rlineto
2 setlinejoin
2 setlinecap
stroke
```

Slika 4 prikazuje izgled stranice opisan tim naredbama.



Slika 4. Linije različitih načina spajanja i vrste rubova

2.4. STAZA

Osim naredbi kojima se definira što će se događati i na koji način sa željenim oblikom, bitan dio koda je staza kojom se taj oblik opisuje. Stazu je naredbama moguće opisati samu, ali se ona na stranici neće pojaviti dok joj se ne doda naredba kojom će se prikazati. Takve se staze mogu sastojati od ravnih linija, Bezierovih krivulja i kružnih isječaka, a osim ispune i obruba, njima se može odrediti i put koji će se prikazati kao maska. Čini ju skup naredbi kojima je određen oblik koji će se kasnije prikazati. Početak staze obično je *moveto* naredba, kojom se oblik postavlja na željeno mjesto na stranici putem x i y točaka na koordinatnom sustavu, a može joj prethoditi i *newpath,* koji briše prethodne staze i stvara novi početak.

2.4.1. Lineto i rlineto

Najjednostavniji oblik koji je moguće definirati je ravna linija. Nova linija na stranicu se smješta unošenjem početne točke *moveto* naredbom, kojom se određuje u kojoj će točki biti početak linije. Nakon toga određuje se točka u koordinatnom sustavu do koje će ta linija biti povučena. Naredbom *rlineto* određuje se relativan pomak točke do koje linija ide u odnosu na točku od koje je počela, te koordinate iz koje ona počinje tada se odnose kao ishodište koordinatnog sustava prema završetku takve linije. Točku u kojoj linija završava moguće je definirati i preko stvarnog položaja u koordinatnom sustavu, upisujući njezine koordinate za x i y počevši od ishodišta stranice. U tom slučaju se

koristi *lineto* naredba. Staza koja čini oblik kvadrata veličine 100 točaka s početkom u ishodištu stranice može se definirati koristeći samo naredbe za dodavanje linija na stranicu na sljedeći način:

```
newpath
0 0 moveto
100 0 lineto
100 100 lineto
0 100 lineto
closepath
fill
```

U tom je primjeru oblik iscrtan linijama, nakon čega je *closepath* naredbom staza zatvorena, a oblik ispunjen.

Slika 5 prikazuje takvu ispunu.

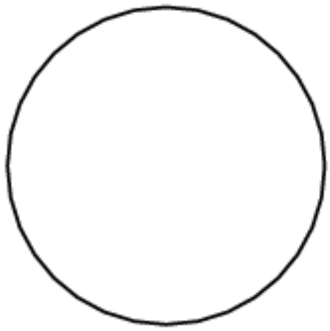
Slika 5. Ispuna staze

2.4.2. Kružni isječci i odsječci

Kod oblikovanja kružnih isječaka nije potrebno dodavati *moveto* naredbu kako bi ih se smjestilo na stranicu budući da skup naredbi kojim se oni definiraju u sebi već sadrži koordinate kojima se smještaju na stranicu, a čine njihovo središte. Naredba kojom se opisuju kružni isječci je *arc*, a osim središta kružnog isječka, za njegovo je oblikovanje potrebno definirati i radijus te početnu i završnu točku kružnice koja ga opisuje, definirane u stupnjevima koje kut od središta s njima zatvara, a ide od 0 do 360. Tako bi se jedna cijela kružnica radijusa 50 točaka s ishodištem na 100 točaka po x i po y koordinati u naredbi pozvala:

```
100 100 50 0 360 arc
stroke
```

Slika 6 prikazuje takvu kružnicu.



Slika 6. Obrubljena kružnica

Ako se na stranici želi prikazati ispunjeni kružni odsječak, primjerice 50 posto sive ispune i radijusa 50 točaka, istome je potrebno u naredbi dodati i *moveto* s koordinatama jednakim njegovom središtu kako bi staza povezala središte s opisanim dijelom kružnice i na taj način dala cijeli kružni odsječak. Naredbe kojima je to moguće postići glase :

```
100 100 moveto
100 100 50 0 90 arc
0.5 setgray
fill
```

Slika 7. Kružni isječak

2.4.3. Bezierove krivulje

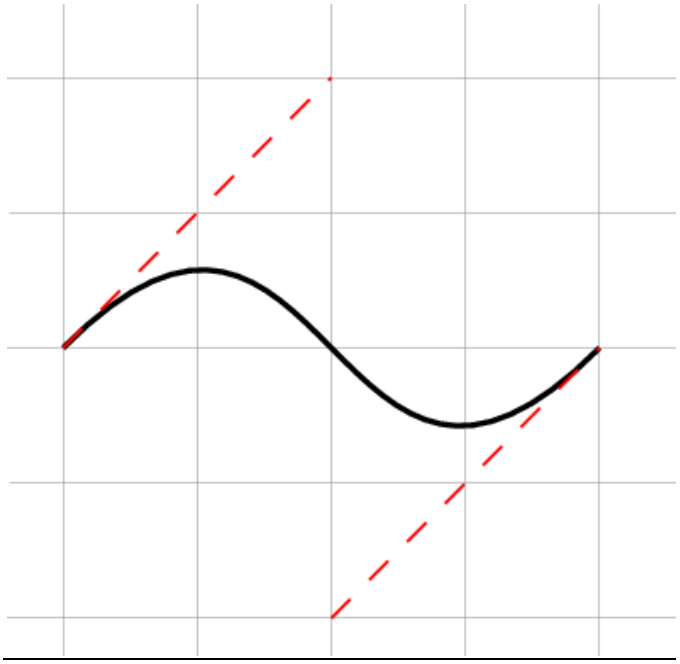
Osim ravnih linija, PostScript omogućuje i prikaz zaobljenih linija i krivulja koje se izvode na principu Bezierovih krivulja. Svaka se od tih linija iscrtava pomoću curveto naredbe koja u sebi sadrži vrijednosti te se u cijelosti ispisuje kao

```
x1 y1 moveto
x2 y2 x3 y3 x4 y4 curveto
```

gdje x1 y1 označavaju koordinate početne točke krivulje, x4 y4 završne točke, a vrijednosti između njih stoje za točke prema kojima one naginju. Kao i kod naredbi za ravne linije, krivulje je moguće na stranici postaviti relativnim položajem u odnosu na prethodnu točku pomoću *rcurveto* naredbe. Primjer koji iscrtava takvu krivulju je:

```
0 0 moveto
100 100 100 -100 200 0 rcurveto
2 setlinewidth
stroke
```

U primjeru na slici 8 ta je krivulja postavljena na mrežu čije su linije postavljene u razmacima od 50 točaka. Krivulja čiji kod je dat u gornjem primjeru prikazana je crnom bojom, a crvene isprekidane linije vode do koordinata koje su korištene u naredbi za krivuju te označavaju zamišljene linije kojima krivulje naginju.



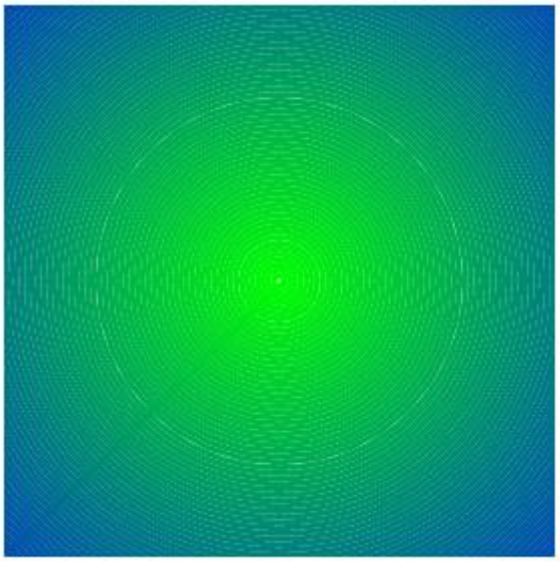
Slika 8. Bezierova krivulja

2.5. MASKE

Ukoliko se na stranici želi prikazati oblik ispunjen nekim uzorkom, moguće je oblik dodati kao masku željenom uzorku. Maska se počinje iscrtavati tako što se početak staze definira naredbom *newpath,* zatim slijedi oblik koji maska opisuje, a taj se oblik zatvara *closepath* naredbom. Kada se oblik odredio, slijedi naredba *clip,* koja zatim sve što se naknadno iscrtava dodaje kao uzorak kojim se željeni oblik popunjava. Kako bi unutar maske ostao samo uzorak koji želimo, a ne sve što nakon *clip* naredbe slijedi, potrebno je nakon dodavanja maske vratiti prethodno stanje *grestore* naredbom. Primjer takve maske prikazan naredbama:

```
newpath
0 0 moveto
100 0 lineto
100 100 lineto
0 100 lineto
closepath
clip
/r 0 def
/g 1 def
/b 0 def
100 {
/r r 1 add def
/b b 0.01 add def
/g g 0.01 sub def
0 g b setrgbcolor
50 50 r 0 360 arc
stroke
} repeat
```

Slika 9 prikazuje masku unutar koje su kao ispuna dodane koncentrične kružnice s promjenom boje od zelene do plave kroz petlju.



Slika 9. Maska od koncentričnih kružnica s promjenom boje

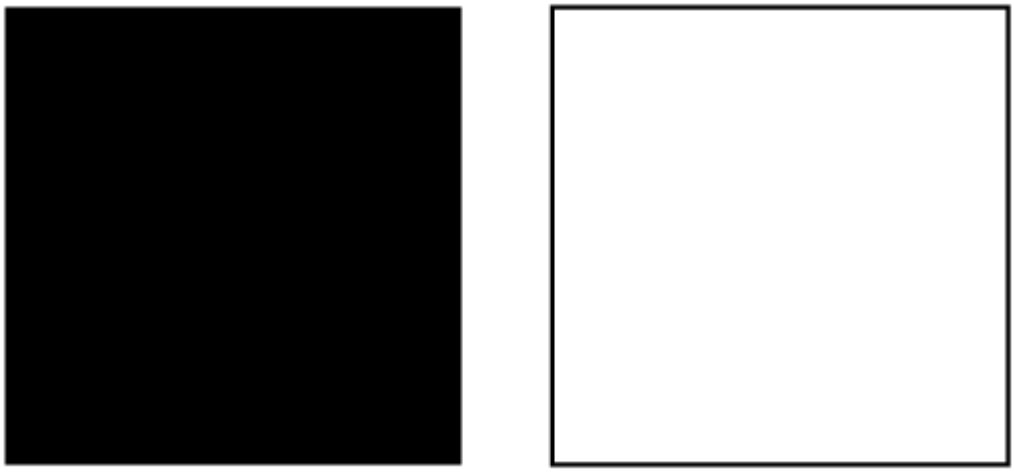
2.6. FILL I STROKE

Kada se naredbama definira staza i način na koji će se ona prikazati, upisuju se naredbe kojima se tako opisana staza prikazuje na stranici. Oblike je tako moguće obrubiti, što se postiže naredbom *stroke* ili ispuniti upisujući *fill* naredbu. Naredbe *fill* i *stroke* u kodu stoje same za sebe i uz njih nije potrebno dodavati vrijednosti budući da služe samo za određivanje što će se sa opisanom stazom dogoditi. Jednom kada se opisani oblik obrubi ili ispuni, odnosno kada se izvrši *fill* ili *stroke* naredba, isti je oblik potrebno ponovno opisati kako bi se ponovno prikazao na stranici. Ako se na stranici želi prikazati oblik koji se ponavlja, to je moguće olakšati spremajući trenutno stanje nakon naredbi kojima je oblik opisan pomoću naredbe gsave te povratak tog stanja nakon svakog puta kada se taj oblik ispuni. Još je jedan jednostavniji način ispunjavanja ponavljajućeg oblika, a to je definiranje oblika kao procedure.

U sljedećem primjeru prikazana je naredba koja opisuje dva kvadrata, jedan ispunjen, a drugi obrubljen crnom bojom:

```
%1
0 0 moveto
100 0 rlineto
0 100 rlineto
100 neg 0 rlineto
```

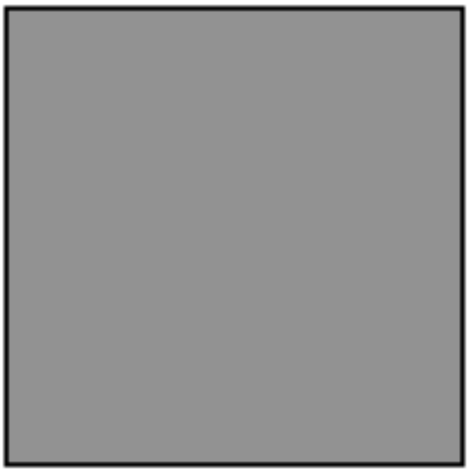
```
closepath
fill
%2
120 0 moveto
100 0 rlineto
0 100 rlineto
100 neg 0 rlineto
closepath
stroke
```



Slika 10. Oblik s ispunom i isti oblik obrubljen

Primjer ispune sivom bojom i obruba istog oblika pomoću *gsave* i *grestore* naredbe koja će u kodu pratiti prijašnje stanje, odnosno stazu:

```
0 0 moveto
100 0 rlineto
0 100 rlineto
100 neg 0 rlineto
closepath
gsave
0.5 setgray fill
grestore
stroke
```



Slika 11. Oblik ispunjen i obrubljen

2.7. VARIJABLE I PROCEDURE

Procedura može biti skup naredbi koji se definira željenim nazivom te se kao takva odvije svaki put kada bi se u kodu zadanim imenom pozvala kao naredba. Početak procedure uvijek je kosa crta, a kada se skup naredbi u njoj sadržan upiše unutar vitičastih zagrada, njezin se završetak označava kao *def*. Ukoliko je kao procedura označen samo jedan broj koji se u kodu želi mijenjati kroz petlje ili zvati drugim imenom, tada vitičaste zagrade u proceduri nisu potrebne, već ju je moguće napisati koristeći samo kosu crtu, željeno ime, broj i naredbu *def* za kraj. Tako bi, na primjer, `/a 10 def` bila naredba koja označava da varijabla a ima vrijednost 10.

2.8. RAČUNSKE OPERACIJE

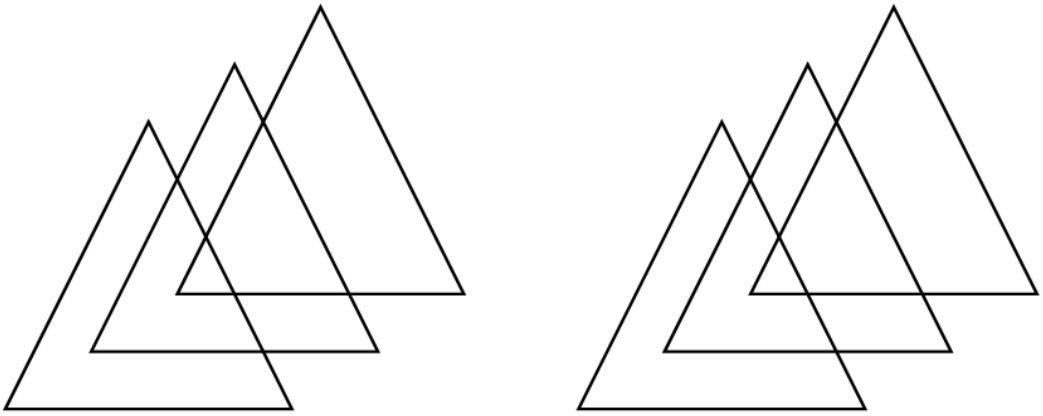
Kada se koriste petlje, takve procedure omogućuju da se brojevi svakim prolaskom kroz petlju mijenjaju. U petlju se tada dodaje još jedna procedura, uz koju se dodaje računska operacija kojom će se taj broj promijeniti. PostScript sadrži računske operacije zbrajanja (*add*), oduzimanja (*sub*), dijeljenja (*div*), množenja (*mul*) i korjenovanja (*sqrt*) te računa arcus tangens (*atan*). Naredbama za računske operacije dodaju se vrijednosti s kojima se one izvršavaju, odvojene samo razmacima, izuzev operacije za korjenovanje koje sadrži samo jednu vrijednost.

2.9. PETLJE

U PostScriptu postoje dvije vrste petlji, a to su *for* i *repeat* petlja. Naredbe koje se za svaku petlju upisuju unutar vitičastih zagradi odvijaju se u zadanom broju ponavljanja. *For* petlja određena je početnom vrijednošću, korakom za koji se ta vrijednost mijenja i

završnom vrijednošću. Te su vrijednosti odvojene razmakom, nakon čega naredbe za ponavljanja kroz petlju slijede unutar vitičastih zagrada i završava naredbom *for*. Za razliku od nje, *repeat* petlja definirana je kao kombinacija vrijednosti za broj ponavljanja izražena u broju ponavljanja, naredbi koje se ponavljaju u petlji unutar vitičastih zagrada i *repeat* naredbe na kraju petlje. Jednak je oblik na stranici tako moguće postići koristeći različite petlje, pri čemu sadržaj koji se nalazi unutar petlje može ostati potpuno jednak.

```
gsave
%for petlja
0 1 2 {
0 0 moveto
100 0 lineto
50 100 lineto
closepath
stroke
30 20 translate
} for
grestore
200 0 translate
%repeat petlja
3 {
0 0 moveto
100 0 lineto
50 100 lineto
closepath
stroke
30 20 translate
} repeat
```



Slika 12. Isti oblik prikazan *for* i *repeat* petljom

2.10. PSEUDOSLUČAJNI BROJEVI

Pseudoslučajni brojevi prikazuju vrijednosti koje se temelje na početnom sjemenu pomoću kojega se zatim određuju okviri unutar kojih će se one prikazivati. Generator slučajnih brojeva postavlja se u kod naredbama i izgleda ovako:

```
123456789 srand
/m {2 31 exp 1 sub} def
/rn {rand m div} def
```

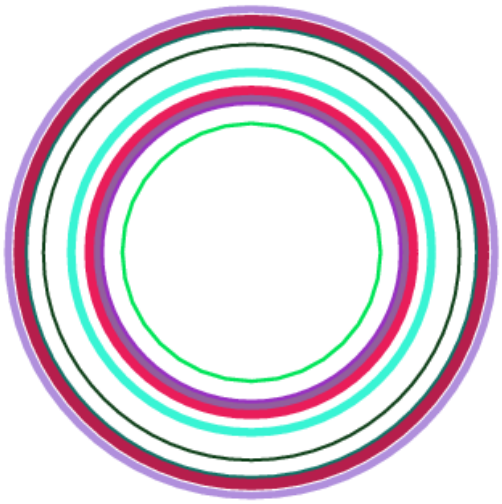
Broj koji stoji uz *srand* naredbu predstavlja sjeme čija promjena utječe na promjenu nasumičnih vrijednosti u definiranom polju. U tom je generatoru *rn* definirana kao varijabla čija je vrijednost slučajan broj od 0 do 1. Pomoću te varijable postavljaju se željeni okviri unutar kojih će se izvući vrijednosti za slučajne brojeve. Takav se okvir postavlja pomoću naredbi za računske operacije, a definira se kao procedura kojom će se ta vrijednost pozivati naredbama kasnije u kodu. [5]

Kako bi se na stranici prikazalo 10 kružnica nasumično određene boje, sa središtem u istoj točci, radijusom veličine 50 do 100 točaka, kojega u kodu prikazujemo kao varijablu s imenom *r,* a debljinom linije kao varijablu *l* veličine 1 do 5 točaka, može se koristiti sljedeći skup naredbi:

```
/r {rn 50 mul 50 add} def
/l {rn 2 mul 1 add} def
10 {
rn rn rn setrgbcolor
```

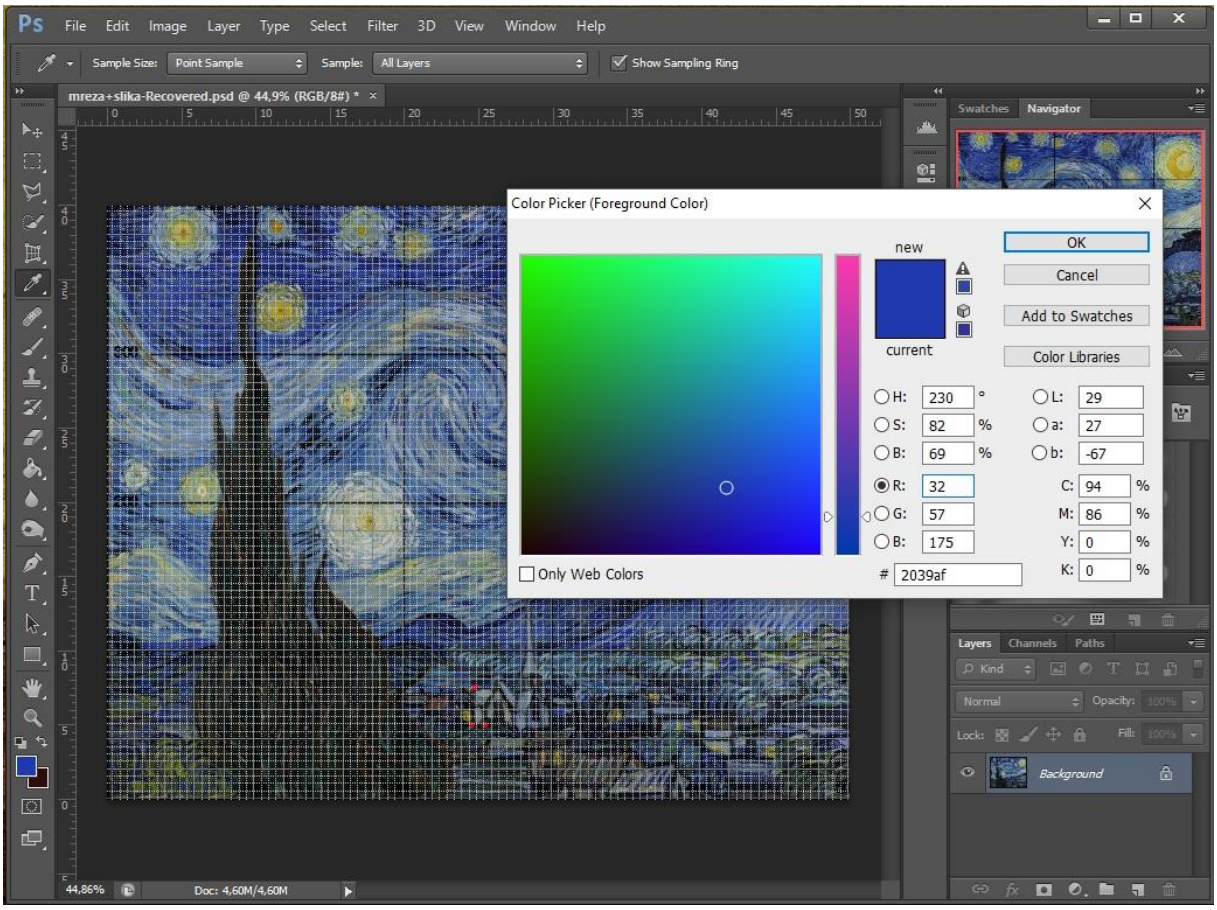
```
100 100 r 0 360 arc
stroke
} repeat
```

Prethodno je u kodu potrebno dodati ranije objašnjen generator slučajnih brojeva, gdje je varijabli *rn* već dodana vrijednost između 0 i 1 te pridodana vrijednostima za boju u RGB sustavima kako bi izbor boje bio potpuno slučajan za svaku kružnicu u petlji. Ovaj niz naredbi program će na stranici prikazati na sljedeći način:



Slika 13. Primjena slučajnih brojeva u definiranju boje i polumjera kružnica

3. EKSPERIMENTALNI DIO



Slika 14. Prikaz korištenja Photoshopa

Izradu ovog završnog rada olakšao je Photoshop, pomoću kojega je lako očitati položaj pojedinih elemenata slike kada se preko nje postavi mreža. Položaj tim elementima potrebno je potom dodati kao koordinate u PostScriptu, čiji koordinatni sustav počinje u donjem lijevom kutu stranice. Photoshop je omogućio i očitavanje vrijednosti boja sa dijelova slike, čije je vrijednosti za sustave boja samo trebalo preračunati u vrijednosti u PostScriptu. Nakon što se odrede koordinate onoga što se u PostScriptu želi prikazati, najčešće se upisuju koristeći *moveto* naredbu kao vrijednosti za x i y, odnosno x y moveto. Program ne može iscrtati oblik ukoliko ne zna na kojoj ga poziciji treba smjestiti, zato je *moveto* početak gotovo svakog niza naredbi. Takav niz naredbi čini samo kostur s kojim je potrebno naknadno definirati što će se događati, odnosno hoće li program iscrtati krug (*fill*) ili samo kružnicu (*stroke*). Prije nego što se naredbom definira što će se sa zadanim oblikom dogoditi, definira se boja u kojoj se on prikazuje, a ukoliko je riječ o stroke naredbi kojom se iscrtava linija koja prati oblik, može se

definirati i oblik linije. Osim debljine, moguće je definirati i to hoće li linija biti konstantna ili isprekidana (*setdash*) te kako će izgledati završetci linije (*setlinecap i setlinejoin*). Isprekidane linije koristile su se u dočaravanju neba krivuljama, kao i u kružnicama koje čine zvijezde, a istima je dodan i zaobljen rub pomoću *setlinecap* naredbe.

Dok je za nebo najviše korištena naredba bila Bezierova krivulja kroz promjene i ponavljanja u petlji, zvijezde su prikazane kružnim isječcima, a oblik stabla opisan je stazom od krivulja te potom popunjen. Koristeći te i brojne druge naredbe moguće je uz prilično vjerno dočarati i najsloženije oblike, a konačno će rješenje završnog rada pokazati koliko vjeran prikaz je moguće postići u prikazu „Zvjezdane noći“ uz poznavanje osnova PostScripta.

Slika 15. Vincent Van Gogh, Zvjezdana noć, preuzeto s http://www.moma.org/learn/moma_learning/vincent-van-gogh-the-starry-night-1889

3.1.NEBO – PETLJE I NASUMIČNE VRIJEDNOSTI

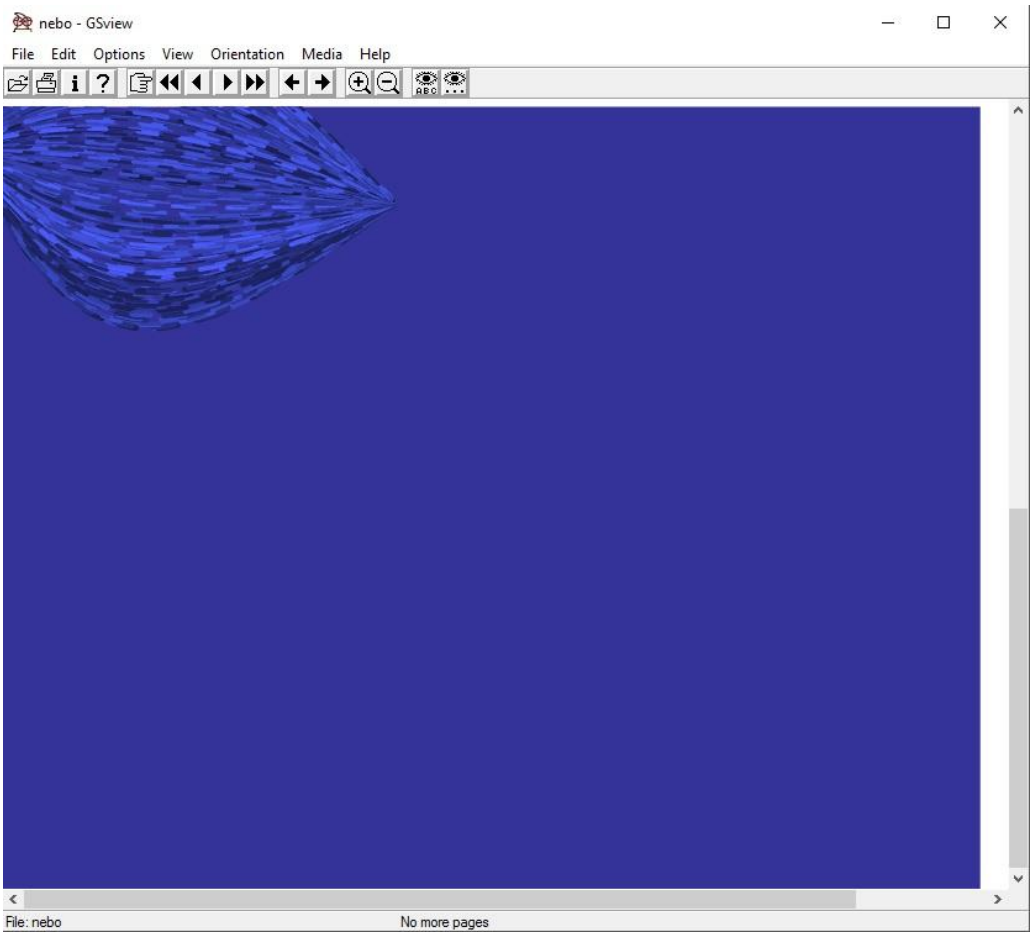
Upravo je nebo najdinamičniji dio Van Goghove slike. Pokušavajući dočarati treperenje zvijezda u mirnoj noći, on je postigao dojam kao da je cijelo nebo živo i grad koji spava izgleda kao da pleše. I dok je u originalnom djelu takve nemirne linije lako bilo dočarati uz nekoliko poteza, iscrtavanje istih u PostScriptu zahtjeva pažljivo mjerenje i računanje. Prvi korak bio je dodavanje plave pozadine na dio stranice na kojem će slika biti iscrtana u omjerima originalne slike, a nakon toga slijedi prikaz neba krivuljama. Iscrtavanje svake krivulje posebno u radu je olakšano korištenjem petlji tamo gdje su one povučene jedna do druge te im je samo dodana promjena vrijednosti na mjestima gdje se one kroz petlju mijenjaju. Petlje se koriste tako da se u zadanom broju ponavljanja ispuniti željena radnja, a u primjeru krivulja koje opisuju nebo očitane su koordinate prve i zadnje Bezierove krivulje u nizu te su se promjena njihovih vrijednosti i pomak odredili brojem ponavljanja u petlji.

Boje koje se u petlji koriste su različite nijanse plave boje, stoga se njihova promjena kroz petlju odredila kao nasumični odabir nijanse boje u željenom rasponu plavih, a raspon boja za svaku je boju definiran drugačije, ovisno o dijelu slike, budući da njihova svjetlina varira. Primjer gornjeg lijevog kuta slike i dijela neba koji je na njemu prikazan izgleda ovako:

```
15031995 srand
/m {2 31 exp 1 sub} def
/rn {rand m div} def

%-...gore-lijevo...-%
/sd {rn 10 mul 8 add} def
/l {rn 3 mul} def
/gly -30 def
/b3 {rn 0.7 mul 0.3 add} def

-20 380 translate

-120 1 100 {
/i exch def
```

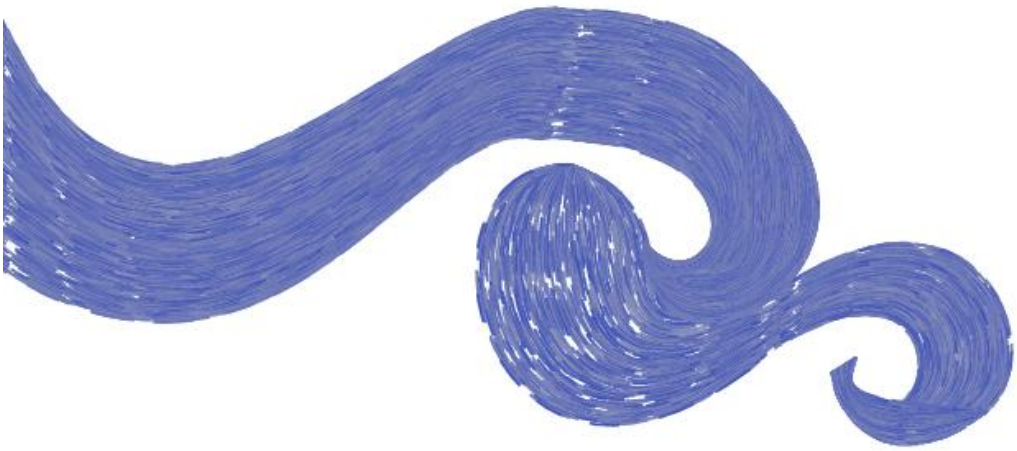
```
/gly gly 0.01 add def
0.65 0.7 b3 sethsbcolor
l setlinewidth
[sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd] sd
setdash
0 0 moveto
60 i 100 i 220 gly curveto
stroke} for
```

U ovom se slučaju radi o *for* petlji, gdje i označava vrijednost koja se kroz petlju svakim ponavljanjem mijenja te raste od -120 do 100. Prethodno su definirane još neke vrijednosti koje se kroz petlju mijenjaju, kao isprekidanost linije (*sd*), vrijednost za plavu boju (*b3*) i debljina linije (*l*), a sve su te vrijednosti definirane kao nasumične u vrijednosti u zadanom rasponu, zbog čega je prethodno potrebno bilo dodati i generator slučajnih brojeva. To omogućuje opuštenije i ne pretjerano pravilan izgled koji više nalikuje originalu. Raspon vrijednosti plavih boja određen je preko Photoshopa te je najveća i najmanja vrijednost za plavu boju u tom rasponu podijeljena s brojem 255. Razlog dijeljenja vrijednosti dobivene na taj način je zato što su u Photoshopu vrijednosti za boje u RGB sustavu prikazane u rasponu od 0 do 255, dok su za svaki sustav boja u PostScriptu one definirane brojevima od 0 do 1. Početak je iscrtavanja *translate* naredbom pomaknut na točku -20 180 u koordinatnom sustavu, a dio koji prelazi granicu slike ne vidi se jer je ona postavljena kao maska unutra okvira, o čemu će se govoriti kasnije u radu. Zbog lakšeg snalaženja među naredbama kojima se prikazuje slika, kroz kod je za svaki element dodan podnaslov čiji naziv ne smeta u prikazu u PostScriptu zbog oznake % koja se ispred njega stavlja.
Gornji lijevi dio slike koji naredbe u ovom primjeru iscrtavaju prikazan je na slici 16.



Slika 16. Početak stranice neba

Velika krivulja svjetlije plave boje na stranicu je dodana kao niz petlji koje se nastavljaju s istim varijablama koje mijenjaju svoje vrijednosti. Sama krivulja prikazana je na slici 17.

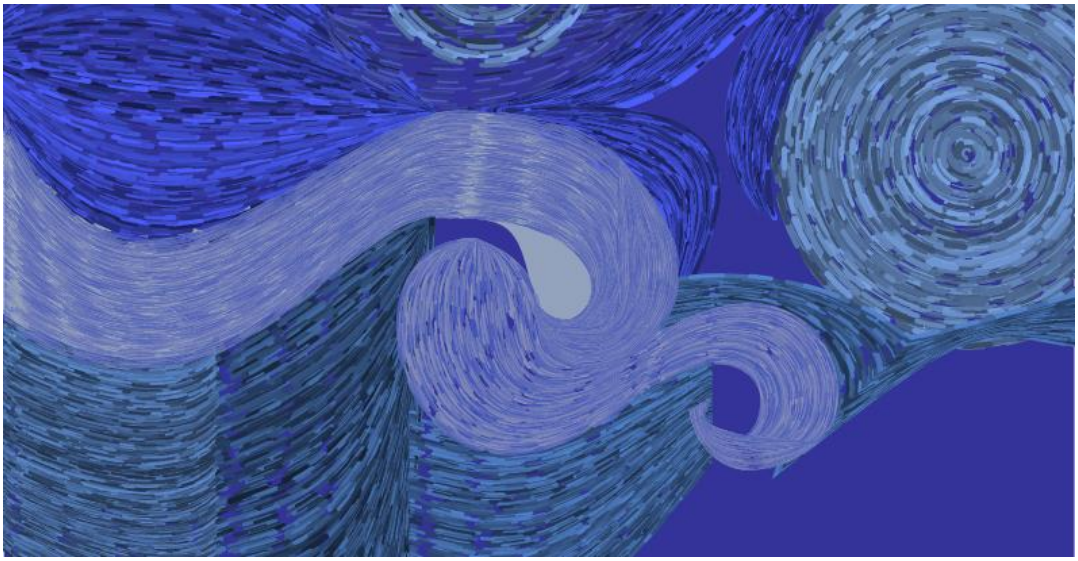


Slika 17. Krivulja sa sredine slike

U kodu ta krivulja izgleda ovako:

```
%%%%OBLAK%%%%%%%%%
15031995 srand
/m {2 31 exp 1 sub} def
/rn {rand m div} def
/sd {rn 10 mul 8 add} def
/ss {rn 0.40 mul 0.25 add} def
%VELIKA KRIVULJA%
/zhnj 350 def
/zhnj1 350 def
/zhnj2 360 def
/zhnj3 210 def
/zhnj4 60 def
/zhnj5 220 def
/zhnj6 250 def
/zhnj7 345 def
/zhnj8 180 def
/zhnj9 380 def
/zzz 365 def
/zzz1 255 def
/zzz2 200 def
500 {
[sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd sd] sd
setdash
0.64 ss 0.72 sethsbcolor
/zhnj1 zhnj1 0.1 sub def
/zhnj2 zhnj2 0.1 sub def
/zhnj zhnj 0.2 sub def
/zhnj3 zhnj3 0.06 sub def
/zhnj4 zhnj4 0.1 add def
/zhnj5 zhnj5 0.025 sub def
/zhnj6 zhnj6 0.05 add def
/zhnj7 zhnj7 0.09 sub def
```

```
/zhnj8 zhnj8 0.1 add def
/zhnj9 zhnj9 0.15 sub def
/zzz zzz 0.17 sub def
/zzz1 zzz1 0.1 add def
/zzz2 zzz2 0.1 add def
0 zhnj moveto
zhnj4 zhnj3 160 zhnj2 zhnj5 zhnj1 curveto stroke
zhnj5 zhnj1 moveto
zhnj9 zhnj7 zhnj6 zhnj8 248 260 curveto stroke
288 327 moveto
zzz 250 zzz1 zzz2 252 248 curveto stroke
} repeat
```



Slika 18. Nebo na kraju

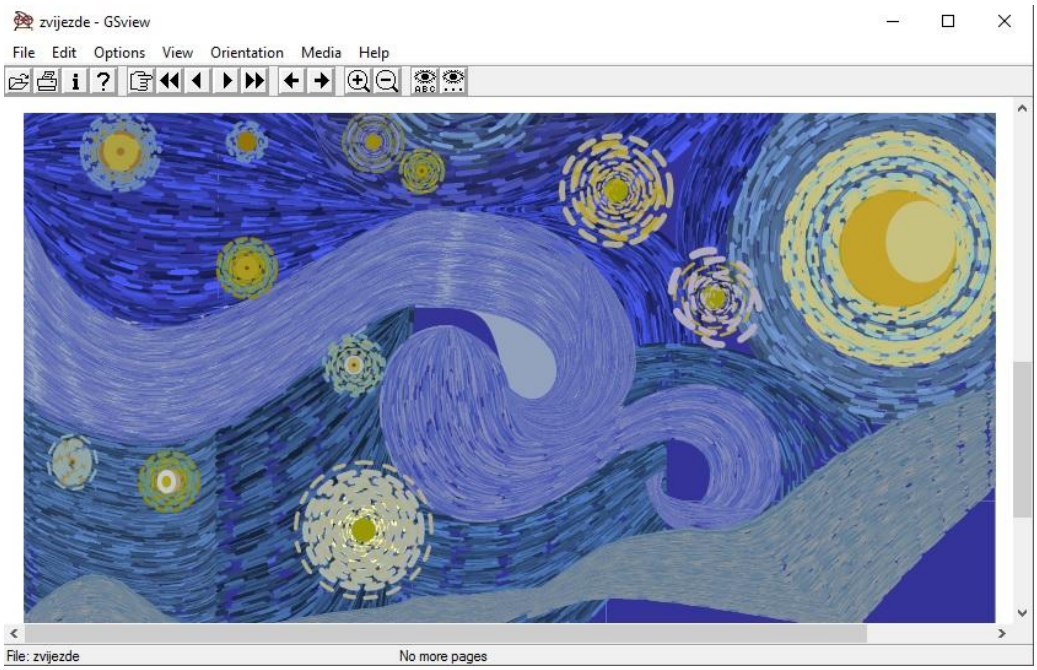
3.2. ZVIJEZDE, MJESEC I ARC

Nakon iscrtavanja neba krivuljama red je došao na zvijezde, koje zbog svog nepravilnog oblika djeluju kao da trepere i predstavljaju pravi izazov za iscrtavanje programiranjem. Prvi i najlakši dio bio je pozicionirati središte svake od njih na stranici, a potom odrediti njihovu boju i polumjer. Svaka je zvijezda opisana na drugačiji način, različitim koncentričnim kružnicama u petlji u kojima je postavljena setdash naredba kako bi njihove linije bile isprekidane, a rub linije je zaobljen pomoću *setlinecap*.

Boje su određene preko Photoshopa, a na nekim su dijelovima kružnice stisnute u elipsu pa su zato korištene i naredbe za oblikovanje cijelog koordinatnog sustava koje djeluju i na objekte koji se u njemu nalaze. *Scale* naredba korištena je za kružnice koje poprimaju oblik elipse.

Kao i kod boja za nebo, tako su i kod opisivanja radijusa kružnica bile korištene nasumične vrijednosti, a korištene su i kao vrijednosti za isprekidanosti linija koje opisuju kružnicu. Primjer koda kojim je prikazana jedna zvijezda:

```
%......zvijezda……10…….%
0.6 0.6 0 setrgbcolor
305 360 6 0 360 arc
fill
gsave
0.72 0.64 0.16 setrgbcolor
305 360 translate
0.7 setlinewidth
[2 2] 0 setdash
15 {
0 0 moveto
0 5 7.5 5 7.5 0 rcurveto
stroke
7.5 0 translate
-1.13 -1.13 scale
30 rotate
} repeat
stroke
grestore
gsave
0.7 0.68 0.54 setrgbcolor
3 setlinewidth
1 setlinecap
305 360 translate
[8 6] 0 setdash
0 1 6 {
0 0 28 0 360 arc
stroke
0.8 0.8 scale
20 rotate
} for
grestore
```



Slika 19. Nebo popunjeno zvijezdama

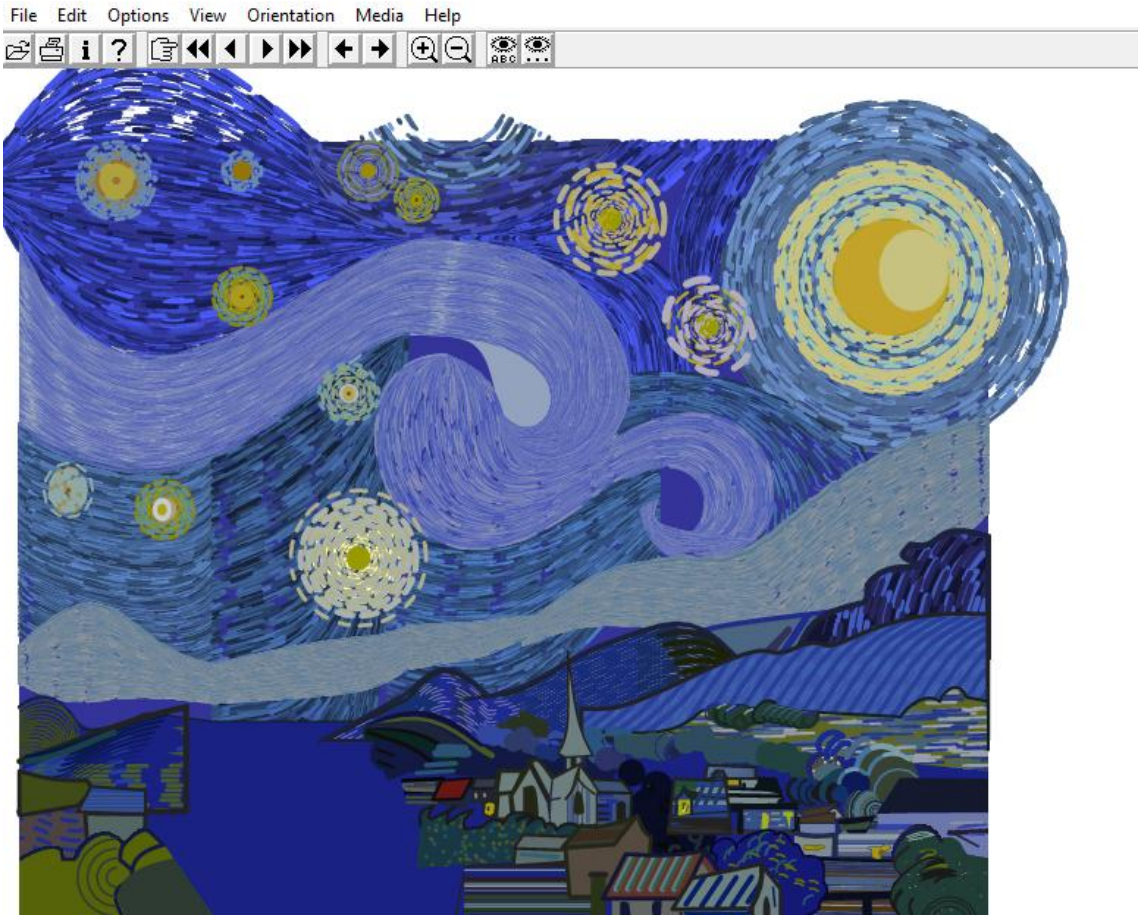
3.3.GRAD I DETALJI

U dočaravanju donjeg dijela slike koja prikazuje grad najteži dio bio je određivanje točnog položaja svakog elementa. Taj dio čine brojne kućice čije su ravne linije prikazane *lineto* i *rlineto* naredbama, a zaobljeni dijelovi opisani su pomoću *curveto* i *rcurveto*. Osim kućica tu su i grmovi koji u nizu mogu biti prikazani kao krugovi s ispunom i dijelovima kružnica koja tu ispunu prati kao sjena. Elementi donjeg dijela slike prikazani su kao oblici s debelim obrubom te je preko svakog elementa nakon što se ispuni, potrebno dodati naredbu *stroke*. Primjer početka koda koji prikazuje crkvu tako izgleda ovako:

```
%crkvica%
248 75 moveto
265 75 lineto
255 50 lineto
246 50 lineto
closepath
0.43 0.5 0.5 setrgbcolor
fill
2 setlinewidth
```

```
248 75 moveto
260 75 lineto
255 64 lineto
246 63 lineto
closepath
245 63 lineto
0.14 0.15 0.09 setrgbcolor
stroke
247 64 moveto
246 50 lineto
255 50 lineto
stroke
3 setlinewidth
255 50 moveto
254 62 lineto
265 80 lineto
274 60 lineto
stroke
2 setlinewidth
274 60 moveto
275 45 lineto
Stroke
```

Slika 20. Prikaz dovršene crkve



Slika 21. Prikaz grada i detalja na stranici

3.4. ČEMPRES – STAZA OD KRIVULJA

Budući da PostScript radi na principu prikaza posljednje dodane naredbe na vrhu stranice, na kraju je tek dodan čempres koji je najbliže prikazan na slici. Cijeli je prikazan kao popunjena staza napravljena od krivulja i potom popunjen, a u kodu izgleda ovako:

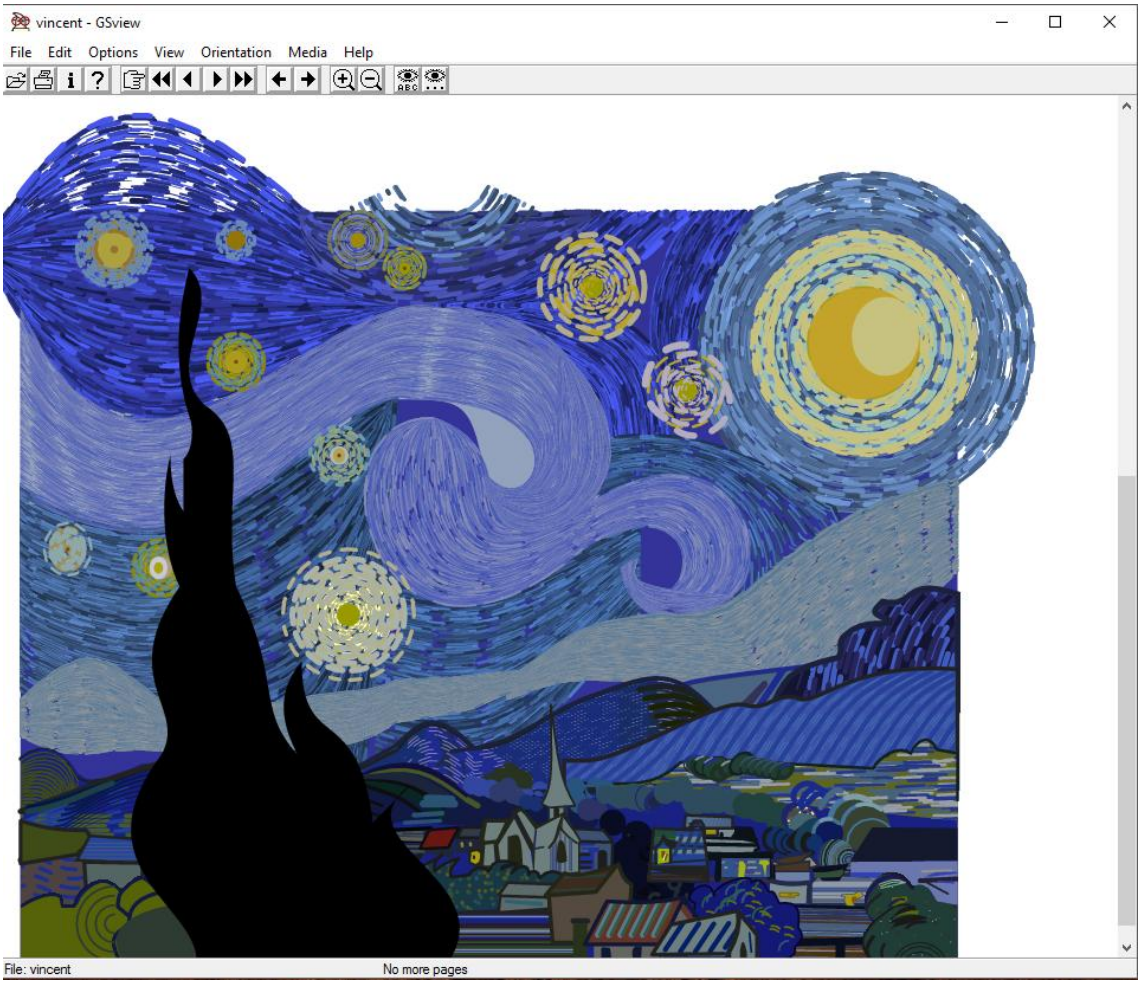
```
0 setgray
94 0 moveto
94 30 30 50 75 110 curveto
90 135 50 150 85 200 curveto
-10 20 -10 50 -5 70 rcurveto
0 -5 0 -20 7 -30 rcurveto
0 28 rlineto
5 5 5 15 4 20 rcurveto
```

```
80 320 85 350 90 370 curveto
110 350 75 320 98 297 curveto
125 280 107 250 115 200 curveto
140 150 125 135 148 110 curveto
-5 10 -15 30 2 50 rcurveto
150 110 180 140 188 74 curveto
20 20 rlineto
-40 -30 50 -50 20 -95 rcurveto
closepath
fill
```

Slika 22. Čempres

3.5.OKVIR KAO MASKA

Budući da su pri oblikovanju neba korištene naredbe za iscrtavanje kružnica, krivulja u petlji i sličnih oblika koje prelaze željene granice na slici, potrebno je cijeloj slici dodati okvir koji će ograničiti prikaz elemenata samo na dio koji opisuje sliku. To je postignuto dodavanjem maske .



Slika 23: Stranica bez okvira

To je postignuto tako što je cijeli kod postavljen kao procedura pod nazivom *vincentvangogh*, odnosno sve je postavljeno unutar vitičaste zagrade procedure.

```
/vincentvangogh {

} def
```

Ta je procedura zatim pozvana kao naredba nakon definiranja maske.

```
newpath

0 0 moveto

500 0 lineto

500 400 lineto

0 400 lineto

closepath

clip

vincentvangogh
```

Slika 24. Krajnji rezultat

4. ZAKLJUČAK

Konačno bi se dalo zaključiti kako PostScriptu ne manjka mogućnosti za primjenu u prikazivanju kompliciranih oblika pa je u njemu moguće prikazati i umjetnička djela iako je za vjeran prikaz potrebno uložiti puno vremena. Sliku je nedvojbeno moguće kreirati do prepoznatljivosti, a od originala će se razlikovati ovisno o redukciji dijelova slike i razrađenosti detalja. PostScript nije namijenjen izradi takvih stranica te je jasno kakao bi se u prikazivanju neke slike u digitalnom obliku više isplatilo korištenje programa koji se bave prvenstveno crtanjem, poput Illustratora, ali u ovom je radu dokazana mogućnost stvaranja umjetnosti programiranjem koje možda daljnjim napretkom tehnologije zamijene čak i digitalni kist kakav takvi programi koriste. Razlika Van Goghove slike i one napravljene u PostScriptu ostaje i u dojmu koji originalna slika ostavlja zbog reljefa nastalog debelim nanosima boje i vidljivih poteza kistom, što još uvijek nije moguće postići na računalu. Imitacijom smjera kretnje krivulja na slici u određenoj je mjeri dočaran nemir i živost originala. Iako je moguće računalom dovoljno vjerno prikazati viziju jednog slikara, unatoč razvoju tehnologije moglo bi se zaključiti kako će slikarstvo na platnu još neko vrijeme ostati nezamjenjivo digitalnom umjetnošću.

5. LITERATURA

1. *https://www.adobe.com/products/postscript/pdfs/PLRM.pdf 28.08.2016.*
2. *http://www-cdf.fnal.gov/offline/PostScript/BLUEBOOK.PDF 28.08.2016.*
3. *http://www-cdf.fnal.gov/offline/PostScript/GREENBK.PDF 28.08.2016.*
4. *Žiljak V., Pap K. (1998). Postscript:Programiranje grafike", FS d.o.o., Zagreb*
5. *Koren, Tajana; Stanić, Nikolina; Rudolf, Maja. Understanding random numbers through postscript // Proceedings of the 10th International Design Conference (Design 2008) Workshop : Design of Graphic Media / Žiljak, Vilko (ur.). Zagreb ; Glasgow : Faculty of Mechanical Engineering and Naval Architecture ; Design Society, 2008. 1487-1490 (predavanje,međunarodna recenzija,objavljeni rad,znanstveni).*