3.vježba – GPJ

Bezierova krivulja	`Px2 Py2 Px3 Py3 Px4 Py4 curveto`
procedura	`/ime {…tijelo procedure…} def`
varijabla	`/ime … def` varijabla sadrži numeričku, tekstualnu vrijednost ili izaz
repeat petlja	`n {…tijelo petlje…} repeat` gdje je n – broj ponavljanja

Bezierova krivulja je definirana sa četiri točke.

Polazna točka Px1, Py1 je definirana kao tekuća točka u kojoj se nalazimo u koordinatnom sustavu, a definira se kroz naredbu moveto ili kao krajnja točka linije, kružnog isječka ili druge krivulje. Ostale tri točke su definirane kao parametri naredbe curveto. Sljedeći primjer pokazuje crtanje krivulje u prostoru poligona koji je omeđen točkama P1(0, 0), P2(0, 100), P3(200, 100) i P4(200, 0):

```
0 0 moveto
0 100 200 100 200 0 curveto
stroke
```

U istom poligonu sa promijenjenim redoslijedom točaka – iscrtat će se krivulja drugačijeg oblika kao što je prikazano u primjeru 01_krivulja.ps. Sve krivulje iz primjera koriste iste krajnje točke poligona sa različitim poretkom, što daje potpuno drugačije krivulje.

Primjer 02_krivulja.ps prikazuje upotrebu naredbe curveto u kontinuiranoj stazi složenoj od više krivulja i linija.

Ako istu stazu želimo ponavljati, mijenjati joj poziciju, boju ili oblik – korisno je spremiti ju kao proceduru te je u programu pozivati kroz ime procedure. Time znatno skraćujemo pisanje koda i činimo ga čitljivijim i pristupačnijim.

```
/list { 5 0 moveto 200 0 200 100 200 200 curveto 200 100 0 200 0 5
curveto 150 120 lineto closepath } def
```

U primjeru 03_procedura.ps je vidljivo kako se definiranjem jedne procedure može brzo konstruirati više staza koje imaju različita svojstva. Koristeći transformacije koordinatnog sustava `translate`, `scale` i `rotate` mijenja se inicijalni oblik, pozicija i veličina staze bez potrebe da se definira više različitih staza.

Ponavljanje u petlji se koristi kada se staza iscrtava mnogo puta te je nepraktično to raditi ručno. U primjeru 04_ponavljanje.ps procedura se ponavlja zadani broj puta i kombinira sa transformacijama koordinatnog sutava. Budući da je staza u proceduri `list` veća nego što želimo smanjit ćemo ju na 30% originalne veličine i translacijom pozicionirati na točku 50, 0. Zatim 20 puta ponavljamo stazu `list` kroz repeat petlju te joj sa svakim ponavljanjem dižemo Y poziciju za 80 točaka.

```
50 0 translate
0.3 0.3 scale

      20 {
         list stroke
         0 80 translate
      } repeat
```

Drugi primjeri u dokumentu 04_ponavljanje.ps na isti način koriste proceduru te ju ponavljaju kroz repeat petlju istovremeno transformirajući njezinu poziciju (translate), scale ili rotaciju.
Kada definiramo boju prije petlje ona će vrijediti za sve oblike koji se ponavljaju u petlji.

U zadnjem primjeru rotirane crvene staze `list` boja je definirana unutar petlje jer se sa svakim ponavljanjem procedure mijenja boja ispune i obruba. Fill i stroke su unutar petlje pa i boja koja se mijenja mora biti unutar petlje.

```
10 { list
     gsave 1 0 0 setrgbcolor fill grestore
     0.5 0 0 setrgbcolor stroke
     36 rotate
     0.95 0.95 scale
   } repeat
```

Uvođenjem varijabli možemo lako kontrolirati željene parametre. Primjer krivulje u 05_varijable.ps koristi varijablu za određivanje broja ponavljanja procedure krivulja putem koje se automatski izračunava kut rotacije. Izraz *brojac = 30* bismo u postscriptu napisali kao:

```
/brojac 30 def
```

Računanje kuta rotacije krivulje definiramo kao *kut = 360 / brojac* odn. 360 dijelimo sa brojem ponavljanja. U postscriptu zapisujemo izraz formule koristeći naredbu *div* koja dijeli dva broja:

```
/kut 360 brojac div def
```

Drugi primjer prema zadanom broju linija i rasponu crta horizontalne linije te automatski računa koliko mora biti Y pomak translacije da bi linije pravilno ispunile zadani raspon.

```
/brojLinija 40 def
/raspon 200 def
/Ypomak raspon brojLinija div def

400 100 translate
brojLinija {
      linija stroke
      0 Ypomak translate
}repeat
```

Prilikom promjene parametara dovoljno je promijeniti vrijednosti varijabli *raspon* i *brojLinija*.

ZADATAK

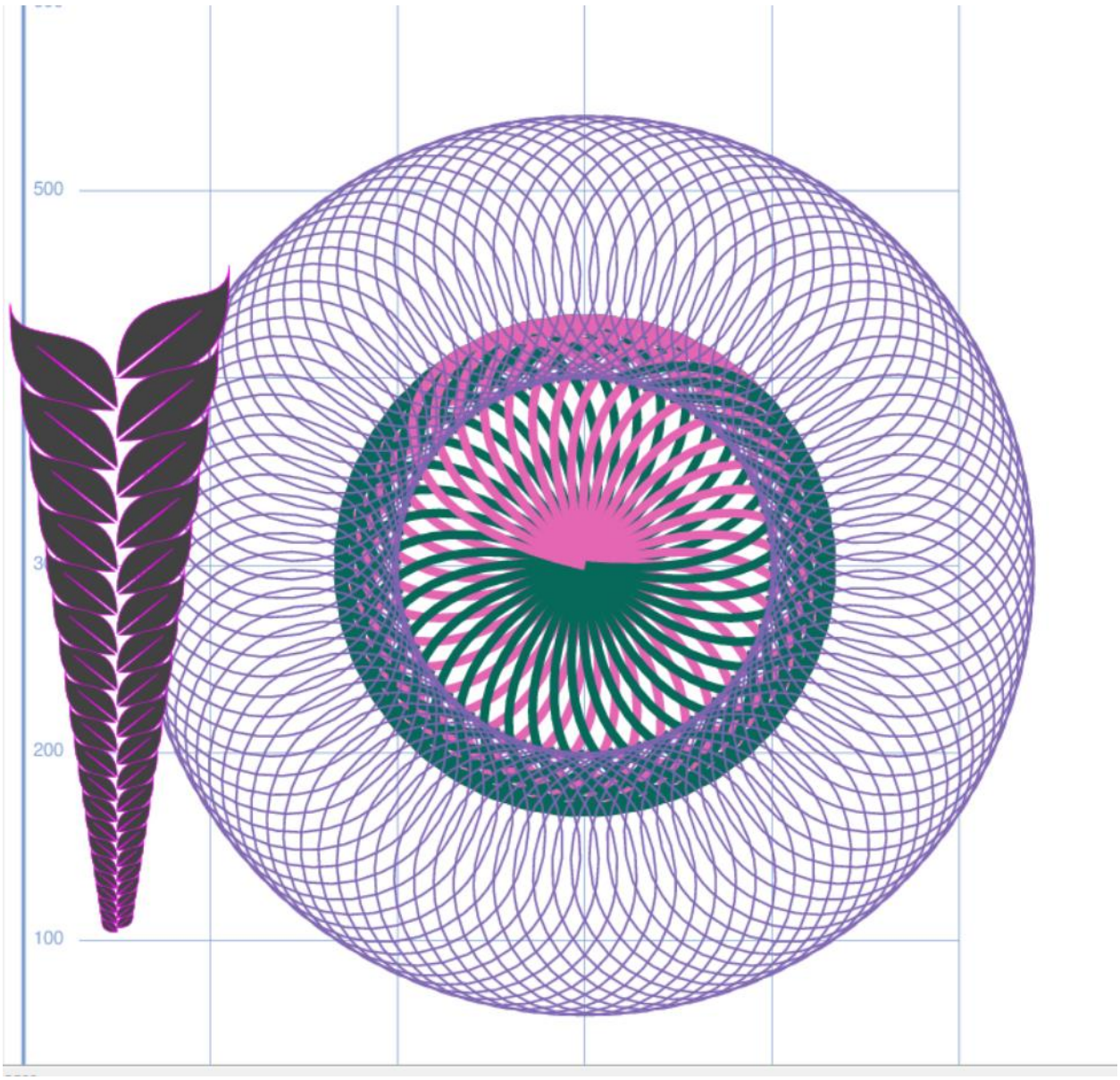
1. Kreirati proceduru "krivulja" koja će sadržavati jednu bezierovu krivulju. Ona se nalazi unutar poligona omeđenog zadanim točkama P1, P2, P3 i P4.
Procedura će se ponavljati u repeat petlji zadani *n* broj puta sa rotacijom koja se računa prema broju ponavljanja tako da u zadnjem iscrtavanju dođe do kuta 360. U istoj petlji se crta i druga procedura "krivulja" koja je zrcaljena u odnosu na prvu krivulju. Boja krivulje je zadana u CMYK sustavu, a boja zrcaljene krivulje koristi iste zadane vrijednosti u RGB sustavu (bez K). Centar rotacije je 300, 300.

2. Proceduru "torus" koje čini kružni isječak. Ponavljati ga u repeat petlji tako da čini torus. Zadan je broj ponavljanja *n1*, veliki i mali radijus te boja obruba torusa (RGB). Kut rotacije se računa na temelju broja ponavljanja.

3. Vlastita procedura složene zatvorene staze sa minimalno dvije krivulje kao u primjeru 04_ponavljanje.ps. Može sadržavati i linije i kružne isječke u kombinaciji sa krivuljama. Ponavljati tu proceduru u petlji proizvoljni broj puta sa translacijom, scale (sa zrcaljenjem ili bez) i/ili rotacijom. Staza mora imati ispunu i obrub obojane različitim bojama u bilo kojem sustavu boja. Sve vrijednosti su proizvoljne.

Primjer zadatka: (individualne zadatke ćete dobiti mailom)

"krivulja"	P1x	0
	P1y	0
	P2x	170
	P2y	10
	P3x	100
	P3y	160
	P4x	0
	P4y	100
broj ponavljanja	n	40
debljina linije	d	4.6
Boja krivulje	C	90
	M	40
	Y	70
	K	20
Torus	Rveliki	100
debljina linije = 1	Rmali	70
broj ponavljanja	n1	100
boja	R	50
	G	40
	B	70
vlastita procedura koja sadrži oblik složene zatvorene staze	obavezno korištenje barem dvije krivulje ponavljanje u petlji translacija, scale i/ili rotate ispuna i obrub proizvoljne boje	


500
200
100