PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **21** (1993/1994)
Številka 6
Strani 324–329

Aleksander Vesel:

# NARIŠIMO KRIVULJO Z RAČUNALNIKOM

Ključne besede: računalništvo.

Elektronska verzija: http://www.presek.si/21/1193-Vesel.pdf

# NARIŠIMO KRIVULJO Z RAČUNALNIKOM

## Uvod

Živimo v ukrivljenem svetu. Da zemlja ni ravna plošča, so vedeli že nekateri davni narodi. Ukrivljena telesa so tudi vsi planeti in zvezde. Piko na i o ukrivljenosti prostora je končno postavil Albert Einstein. Tudi stvari okoli nas so bolj ali manj ukrivljene. Vsaj približno ravne in oglate so predvsem stvari, ki jih je ustvaril človek. Pa tudi glede tega se je že marsikaj spremenilo. Dandanes prevladujejo v industrijskem oblikovanju nežne, zaobljene oblike. Pomislimo samo na najnovejše znamke osebnih avtomobilov!

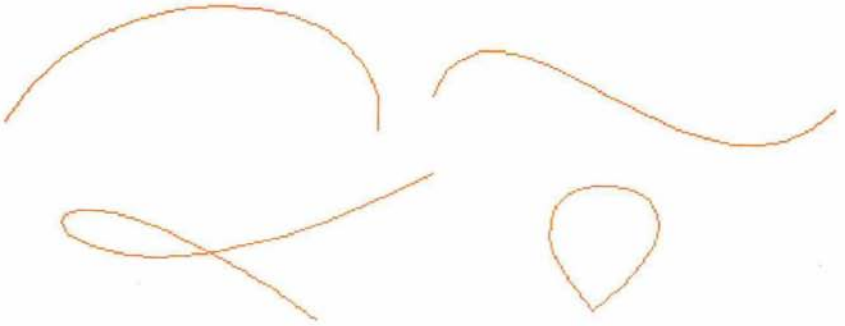
Industrijski oblikovalci pri svojem delu ne morejo več shajati brez računalniške podpore. Na tržišču so številni programski paketi za računalniško podprto načrtovanje (Computer Aided Design) in računalniško podprto risanje (Computer Aided Drafting). Za oba pojma se je uveljavila kratica CAD. Programi za računalniško podprto risanje so preprostejši. Njihov izhod je risba na datoteki ali na papirju. Na osebnih računalnikih sta zelo razširjena predvsem programa *Coreldraw* in *Paintbrush*. Programi za računalniško podprto načrtovanje zmorejo še marsikaj več. Dodane so jim predvsem stvari, ki so pomembne za izdelavo načrtovanega izdelka. So tako obsežni, da jih na osebnih računalnikih še vedno le redko srečamo. Program, ki deloma že sega v to kategorijo, je na primer *Autocad*.

## Aproksimacije in interpolacije

Mnogi ste se verjetno že srečali s programom Paintbrush, ki teče v grafičnemu okolju Windows. Program omogoča risanje ravnih črt, krogov, štirikotnikov, itd. Med temi osnovnimi gradniki so tudi ukrivljene črte, ki jih vnašamo v risbo na čisto poseben način. Najprej med dvema točkama potegnemo ravno črto, nato pa to črto "krivimo", tako da vnesemo še dve dodatni točki. Bolj ko sta dodatni točki oddaljeni od črte, bolj se črta ukrivi. Program je tako prijazen, da se lahko hitro naučimo na različne zanimive načine oblikovati krivuljo. Nekaj primerov je na sliki 1.

Verjetno marsikoga med vami zanima, kako da program ukrivi krivuljo prav na takšen način. Na začetku povejmo, da so krivulje v programskih paketih večinoma predstavljene analitično. To pomeni, da so opisane z (realnimi) funkcijami. Krivulje, ki jih uporabljamo v programih za računalniško podprto risanje oziroma oblikovanje, delimo v dve skupini. Krivuljo, ki gre

skozi vse podane točke, imenujemo **interpolacijska krivulja**. Če gre krivulja samo skozi nekatere podane točke, drugim pa se samo približa, jo imenujemo **aproksimacijska krivulja**.



Slika 1. Krivulje v programu Paintbrush.

### Bézierova krivulja

Izdelovalci programske opreme ne govorijo radi o tem, kakšne analitične opise krivulj uporabljajo. Ker gre v programu Paintbrush krivulja le skozi začetno in končno točko, pravimo, da gre za aproksimacijsko krivuljo. Na podlagi števila potrebnih vnešenih točk in obnašanja krivulje pa lahko sklepamo, da gre za **kubično Bézierovo krivuljo**. Ime je dobila po Pierru Bézieru, ki jo je prvi uporabil pri načrtovanju avtomobilov v francoski tovarni Renault leta 1972. (Danes je že znano, da je Bézierovo krivuljo neodvisno in celo pred Bézierom razvil tudi de Casteljau v konkurenčni tovarni Citroën. Ker pa svojih raziskav ni nikoli objavil, je vso slavo pobral njegov rojak.) Bézierova krivulja je le ena iz velike množice krivulj, ki se uporabljajo v računalniško podprtem načrtovanju oziroma risanju. Zaradi svoje preprostosti je še vedno izredno popularna in je doživela že številne spremembe in izboljšave.

V nadaljevanju sestavka bomo najprej spoznali nekaj najnujnejše teorije. Nestrpnejši bralci pa se lahko takoj lotijo tudi poglavja z računalniškim programom.

### Analitični opisi krivulj

Večina bralcev se je z nekaterimi krivuljami gotovo že srečala. Največkrat so krivulje opisane **eksplicitno**. V ravnini to pomeni, da je spremenljivka $y$

podana kot funkcija neodvisne spremenljivke $x$. Kot primer navedimo enačbo parabole $y = ax^2+bx+c$ in enačbo kubične krivulje $y = ax^3+bx^2+cx+d$.

Verjetno bralci poznajo tudi **implicitni opis**. To pomeni, da je krivulja opisana s funkcijo dveh spremenljivk, na krivulji pa so natanko tiste točke, pri katerih ima funkcija vrednost 0. Lep primer je enačba krožnice s središčem v točki $(a, b)$ in s polmerom $r$: $(x-a)^2+(y-b)^2-r^2=0$.

Niti eksplicitni niti implicitni opis pa nista najbolj primerna za računalniško obravnavo. Implicitni opis je nekoliko "neroden" za realizacijo, eksplicitni pa ne omogoča opisovanja krivulj z večkratnimi vrednostmi. To so tiste krivulje, ki jih vzporednica z osjo $y$ seka večkrat. Na sliki 1 vidimo tudi takšna primera.

Idealnemu opisu se najbolj približa **parametrični opis**. Pri tem opisu sta koordinati $x$ in $y$ funkciji neodvisne spremenljivke $u$ z intervala $I$. Interval $I$ je ponavadi normiran, torej $I = [0, 1]$. Spremenljivki $u$ pravimo tudi **parameter**. Z drugimi besedami to pomeni, da se koordinati $x$ in $y$ spreminjata, ko spreminjamo parameter $u$, pri čemer lahko parameter $u$ zavzame vrednosti med 0 in 1. Včasih želimo poudariti, da sta koordinati $x$ in $y$ odvisni od parametra $u$. Tedaj namesto $x$ in $y$ pišemo $x(u)$ in $y(u)$. Kot primer si poglejmo, kako lahko enačbo krožnice podamo parametrično:

$$x = a + r\cos 2\pi u, \quad y = b + r\sin 2\pi u, \quad u \in [0, 1].$$

Funkciji sinus in cosinus, ki nastopata v zgornjem opisu, so mlajši bralci verjetno spoznali pri pouku fizike. Znani pa sta tudi računalnikarjem, saj sta vgrajeni praktično v vsak programski jezik. Ti dve funkciji sta časovno požrešni, torej takšni, ki za izračun zahtevata precej računalniškega časa.

Polinomske funkcije nižjega reda so časovno precej bolj pohlevne. Izkaže se, da za risanje večinoma zadostuje, če sta koordinati točk na krivulji kubični funkciji parametra $u$. Želimo torej imeti funkciji naslednje oblike:

$$x(u) = a_3u^3 + a_2u^2 + a_1u + a_0 \textbf{ in } y(u) = b_3u^3 + b_2u^2 + b_1u + b_0.$$

Koeficienti $a_i$ in $b_i$ so odvisni od položaja in predvidene oblike krivulje. Spomnimo se spet programa Paintbrush. Položaj in obliko krivulje smo določili s štirimi izbranimi točkami $T_0$, $T_1$, $T_2$ in $T_3$. Te točke bomo v nadaljevanju imenovali **kontrolne točke**. Podali jih bomo s pari koordinat $T_0 = (x_0, y_0)$, $T_1 = (x_1, y_1)$, $T_2 = (x_2, y_2)$ in $T_3 = (x_3, y_3)$. Krivulja naj gre skozi prvo in zadnjo točko, drugima dvema pa se lahko samo približa.

**Povezovalne funkcije kubične Bézierove krivulje**

Funkciji $x(u)$ in $y(u)$ lahko preoblikujemo, tako da je viden vpliv kontrolnih točk:

$$x(u) = F_0(u)x_0 + F_1(u)x_1 + F_2(u)x_2 + F_3(u)x_3,$$

$$y(u) = F_0(u)y_0 + F_1(u)y_1 + F_2(u)y_2 + F_3(u)y_3.$$

Funkcije $F_i$ imenujemo **povezovalne funkcije** krivulje. Povezovalne funkcije so v splošnem kubični polinomi, saj smo predpostavili, da sta funkciji $x(u)$ in $y(u)$ polinoma tretje stopnje.

Bézier je za povezovalne funkcije izbral Bernsteinove polinome. Ker nas zanima kubična krivulja, vzamemo za povezovalne funkcije Bernsteinove polinome tretje stopnje:

$$F_i(u) = B_{i,3}(u) = \binom{3}{i} u^i(1-u)^{3-i} = \frac{3!}{i!(3-i)!} u^i(1-u)^{3-i}.$$

Na ta način dobita funkciji $x(u)$ in $y(u)$ naslednjo obliko:

$$x(u) = (1-u)^3x_0 + 3u(1-u)^2x_1 + 3u^2(1-u)x_2 + u^3x_3$$

in

$$y(u) = (1-u)^3y_0 + 3u(1-u)^2y_1 + 3u^2(1-u)y_2 + u^3y_3.$$

Ker je $x(0) = x_0$ in $y(0) = y_0$ ter $x(1) = x_3$ in $y(1) = y_3$, se bo naša krivulja začela v točki $T_0$ in končala v točki $T_3$.

**Računalniški program**

Povedano zadostuje, da napišemo računalniški program, ki nariše kubično Bézierovo krivuljo glede na štiri podane kontrolne točke. Program je napisan v programskem jeziku pascal, narečje turbo pascal 7.0. Uporabljeni so tudi podprogrami iz pripadajoče grafične knjižnice (enota *Graph*). V tem sestavku je predstavljen samo najpomembnejši podprogram z imenom *KubicniBezier*. Vhod v podprogram je tabela s koordinatami štirih kontrolnih točk (njena deklaracija je podana pred podprogramom). V podprogramu je uporabljena tabela binomskih koeficientov *Binom3* in funkcija *potenca*, ki jo bodo bralci gotovo znali sprogramirati tudi sami.

Manj poučene bralce naj spomnimo, da zaslon računalnika sestavlja pravokotna mreža majhnih pik, ki jim lahko določamo barvo. Mreža je pri mnogih osebnih računalnikih velikosti 640 × 480. Krivulje tako na zaslonu računalnika nikoli ne moremo povsem natančno narisati, ampak jo vedno le bolj ali manj dobro aproksimiramo. Nam bo zadostovalo, če bomo krivuljo aproksimirali z lomljeno črto (lomljenko).

Konstanta *gostota*, ki jo uporabljamo v podprogramu, pove, s koliko ravnimi odseki bomo ponazorili krivuljo. Če aproksimiramo krivuljo s petdesetimi odseki (tako kot je v podprogramu), to pomeni, da razdelimo interval parametra *u* na petdeset (enako dolgih) delov. Podprogram poišče točke na krivulji za zaporedne vrednosti parametra in zaporedne točke poveže z ravno črto (klic podprograma *LineTo*). Primeri štirih računalniško generiranih krivulj so na sliki 2. Pripadajoče kontrolne točke krivulj so označene s krogci.

```
const
  Binom3: array[0..3] of integer = (1,3,3,1); { tabela binomskih koeficientov }
type
  KonTocka = record x,y: integer; end;                    { kontrolna točka }
  KonTocke = array[0..3] of KonTocka;                  { tabela kontrolnih točk }

procedure KubicniBezier(T: KonTocke);
{ Podprogram nariše kubično Bezierovo krivuljo. }
{ Štiri kontrolne točke so shranjene v tabeli T. }
  const
    gostota = 50;                          { gostota aproksimacijske lomljenke }
  var
    korak,                                  { odsek aproksimacijske lomljenke }
    i: integer;
    u,                                                   { parameter krivulje }
    B,                                    { vrednost Bernsteinovega polinoma }
    px,py: real;                                         { točka na krivulji }
begin
  MoveTo(T[0].x,T[0].y);                      { začetek v prvi kontrolni točki }
  for korak:=1 to gostota do begin
    px := 0;  py := 0;                          { inicializacija točke na krivulji }
    u := korak/gostota;                                  { vrednost parametra }
    for i:=0 to 3 do begin                  { izračun Bernsteinovega polinoma }
      B := Binom3[i]*potenca(u,i)*potenca(1-u,3-i);
      px := px+T[i].x*B;  py := py+T[i].y*B;
                                                  { prišteje vpliv kontrolne točke }
    end;
    LineTo(round(px),round(py));              { nariše črto do izračunane točke }
  end;
end; {KubicniBezier}
```

Slika 2. Generirane krivulje.

### Zaključek

Pred klicem podprograma je potrebno vnesti kontrolne točke. Zanimivo je opazovanje vpliva **dvojnih** ali **trojnih** točk. To pomeni, da zaporedoma podamo dve ali tri enake točke (primer je zadnja krivulja na sliki 2). Pred izrisom krivulje je treba preklopiti računalnik v grafični način delovanja, kar gotovo ne bo problem za tiste, ki ste že napisali kakšen grafični program v pascalu.

Podprogram je napisan za štiri podane kontrolne točke, nobene ovire pa ni za posplošitev parametričnega opisa Bézierove krivulje na $n$ točk. V tem primeru postanejo povezovalne funkcije Bernsteinovi polinomi $n$-te stopnje. Zaradi precejšnje časovne zahtevnosti računanja vrednosti polinomov višjih stopenj pa se izogibamo povezovalnim funkcijam, katerih red je višji od 3. Problem aproksimacije z velikim številom kontrolnih točk zaradi tega ponavadi rešujemo tako, da "zlepimo" skupaj več kubičnih krivulj.

*Aleksander Vesel*

## V TRGOVINI – Rešitev s str. 296

Očitno se ženi Miha Mah, torej mora nevesta biti Tina Čeh. Škrjanc mora biti čistilec (ker je že poročen). Tretji moški, Klemen Klemen, je kupec. Poslovodja je torej Eva Kralj, Tina Čeh je blagajničarka, prodajalec pa Miha Mah.

*Neža Mramor-Kosta*