FRAKTALI – ZNANOST ALI UMETNOST

INFORMATICA 4/89

Keywords: fractal, iterated function method, Julia set, Mandelbrot set

Jasna Đonlagić, Nikola Guid
Tehnična fakukteta Maribor

V naravi je veliko objektov in pojavov, kot n.pr. drevesa, gore, oblaki, pretok tekočin, rast populacije, oblika možganskih gub ter podobni pojavi, ki iz določenega reda preidejo v nered in ki jih ne moremo matematično predstaviti z običajnimi orodji, lahko pa jih s fraktalno geometrijo oz. fraktali. V članku obravnavamo naravne in geometrične fraktale, metodo iterativnih funkcijskih sistemov (IFS) in nakažemo uporabo formalnih jezikov. Na koncu izdelamo algoritme za naslednje vrste fraktalov: Juliovo množico, Mandelbrotovo množico in fraktale določene po metodi IFS.

ABSTRACT

For some natural objects like trees, mountains, clouds, flow of liquids, population growth, surface of brain etc., which turn from order into chaos, it is impossible to describe all mathematical data, but it is possible to present them with fractal geometry or fractals. The purpose of this article is to present natural and geometrical fractals, iterated function method (IFS), and use of the formal languages in the fractal theory. At the end we show some algorithms for creating the following fractals: Julia set, Mandelbrot set, and fractals produced by IFS method.

1. UVOD

Beseda fraktal je latinskega izvora (fractus = zlomljen), torej naj bi spominjala na lomljenje, drobljenje. S fraktalno teorijo se je pričel ukvarjati Benoit B. Mandelbrot 1980. leta [6], čeprav so matematične osnove za nastanek te teorije ustvarili že mnogo prej P. F. Verhulst, Gaston Julia, Pierre Fatou, Adrien Douady in drugi [7].

Fraktale delimo na dve osnovni skupini, in sicer na: naravne in geometrične.

Članek obsega opis naravnih in geometričnih fraktalov ter uporabo formalnih jezikov na področju fraktalne teorije. V povezavi z naravnimi fraktali smo opazovali Verhulstov dinamični proces ter Juliove in Mandelbrotove množice. Pri geometričnih fraktalih smo podali metodo IFS in opisali nekatera pravila za generiranje geometričnih fraktalov. Prav tako smo razvili tri algoritme za generiranje fraktalnih slik.

2. NARAVNI FRAKTALI

Naravne fraktale lahko zasledimo na področju geografije, biologije, biokemije in fizike, ko želimo z njimi upodobiti naravne pojave, kot so na primer poplave rek, pretok tekočin, obliko možganskih gub, vaskularni sistem, vreme itn.

Ideja za nastanek naravnih fraktalov izvira iz biologije, ko je P. F. Verhulst 1845. leta definiral zakon rasti populacije [7]. Zagovarjal je trditev, da lahko določena populacija narašča tako dolgo, dokler ne doseže svojega razsežnostnega maksimuma X. Če le tega prekorači, velikost populacije pade. Potrebno je bilo več kakor sto let, da so dokazali vse nejasnosti te trditve. Prišli so do zanimivih rezultatov :

Če je faktor rasti majhen, se velikost populacije uravnava sama po sebi in zmeraj ostaja znotraj optimalne vrednosti X. To pomeni, da populacija narašča, kadar je pod optimalno vrednostjo X, in pada, kadar je nad optimalno vrednostjo X. V primeru, da je faktor rasti velik, takšen, da se populacija poveča za več kakor 200 %, pa optimalne vrednosti X ni mogoče več doseči.

Porodi se vprašanje, ali je rast populacije sploh kdaj tako velika? Seveda! Človeške populacije ne naraščajo tako hitro, toda pri nekaterih insektih tak pojav ni nenavaden. Ugotovili so, da pri 245 % povečanju populacije nastanejo oscilacije okoli optimalne vrednosti X z velikostjo periode 2,4,8,16,..., dokler pri 257 % ne preidemo v zmedo.

Kako si razlagamo besedo zmeda? To preprosto pomeni, da delovanja sistema ne moremo več nadzorovati, da je ušla izpod naše kontrole.

Najzanimivejša točka Verhulstovega dinamičnega procesa pa ni zmeda sama po sebi, temveč dogodek, ki spremeni red v nered. Verhulst je to trditev matematično dokazal. Z x_0 označi začetno velikost populacije, z x_n pa velikost populacije po n letih. Jakost rasti populacije R definira relativni prirastek populacije na leto:

$$R = \frac{x_{n+1} - x_n}{x_n}$$

Nato uvede določene predpostavke:

- Populacija lahko naraste do določenega maksimuma X (maksimum normirajmo, tako da je X=1)
- R se naj spreminja glede na velikost populacije in naj bo linearna funkcija od r (r imenujemo parameter rasti in r>0): $R=r(1-x_n)$

Ob teh predpostavkah dinamični zakon rasti preide v naslednjo obliko:

$$x_{n+1} = f(x_n) = (1+r)x_n^2 - rx_n$$

Ločimo dve stanji, pri katerih populacija ne spreminja velikosti: $x_0 = 0$ in $x_0 = 1$. Zanima nas stabilnost oz. nestabilnost teh dveh stanj. Pri $x_0 = 0$ ne pride do rasti populacije, saj nimamo s čim začeti. To stanje smatramo za nestabilno, saj že pri $0<x_0<<1$ nastopi rast populacije:

$$x_1 \cong x_0 - rx_0$$

$$x_2 \cong x_1 - rx_1$$

...itn.

Zaporedje $x_0, x_1, x_2, \ldots$, narašča, dokler ne doseže velikosti 1. Ali je stanje $x_0 = 1$ stabilno? Da bi prišli do odgovora, opazujmo razliko $\delta_n = x_n - 1$! Dinamični zakon nam omogoča izračun x_{n+1}, torej lahko določimo tudi naslednji δ_{n+1}.

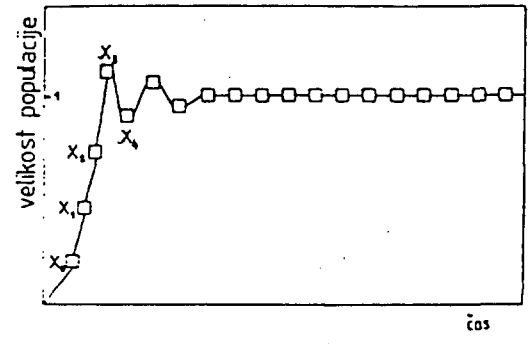
$$\delta_{n+1} \cong \delta_n(1-r)$$

Težimo za tem, da se x_{n+1} stabilizira pri 1, zato mora δ_{n+1} limitirati proti 0.

$$\lim_{n\to\infty} \delta_{n+1} = 0 \quad \Rightarrow \quad \lim_{n\to\infty} \delta_0(1-r)^n = 0$$

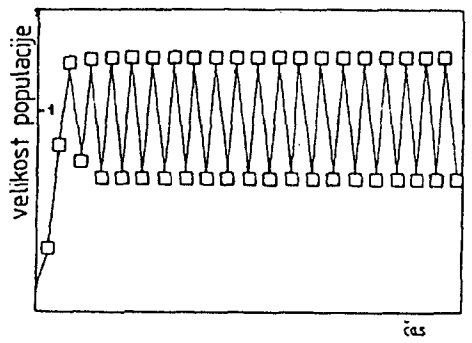
To bo veljalo, kadar bo $|\delta_{n+1}| < |\delta_n|$ (z x_{n+1} se bomo približevali k 1), če je $0<r<2$. Ali z drugimi besedami: Stanje $x_0 = 1$ je stabilno, če r leži v mejah med 0 in 2 ter nestabilno, če je r>2.

Oglejmo si primer, pri katerem postavimo $x_0 = 0.1$ in r=1.8 (slika 2.1).

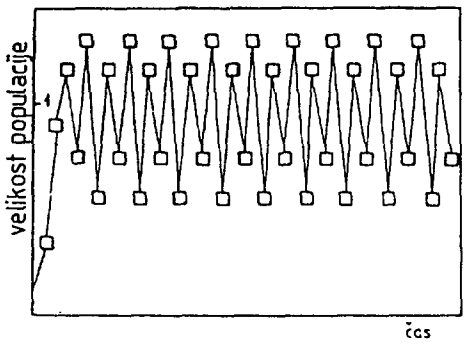


Slika 2.1 Rast populacije pri r=1.8 in $x_0 = 0.1$

Vidimo, da velikost populacije x sprva narašča, saj je pod maksimalno vrednostjo X=1. V četrtem koraku pride do prekoračitve. Zaradi tega se velikost populacije zmanjša in pade pod 1. To povzroči ponovno rast in takoj za tem ponoven padec itn., dokler se dokončno ne umiri pri stabilni vrednosti X=1. Zanimivejši položaj nastane, kadar je r>2. Zato raziščimo primer, ko postavimo $x_0 = 0.1$ in r=2.3 (slika 2.2) ter $x_0 = 0.1$ in r=2.5 (slika 2.3).

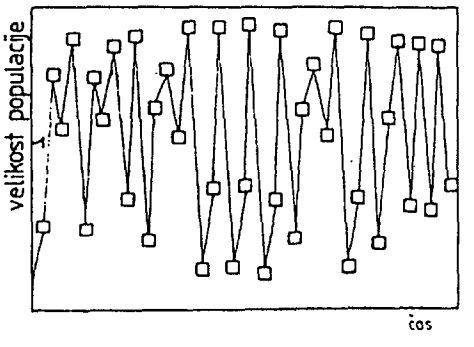


Slika 2.2 Rast populacije pri r=2.3 in x_0= 0.1



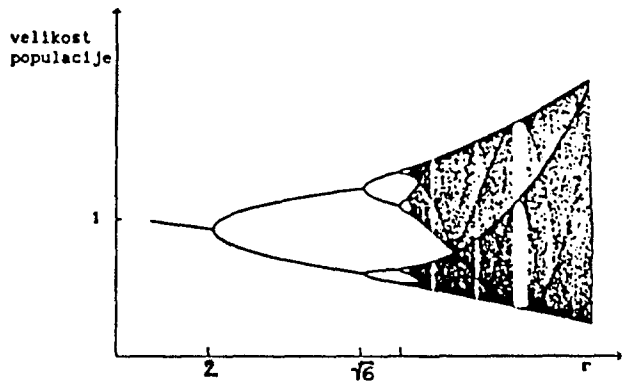
Slika 2.3 Rast populacije pri r=2.5 in x_0= 0.1

Ugotovimo, da pride do periodične oscilacije med nivoji. V prvem primeru dobimo periodo 2, v drugem pa 4. Če bi nadaljevali s postopkom, bi dobili periodo 8,16,32,..., dokler pri r=2.57 ne bi prešli v zmedo in periode ne bi bilo več mogoče določiti (slika 2.4).



Slika 2.4 Rast populacije pri r=2.57 in x_0= 0.1

V nadaljevanju si lahko zastavimo zanimivo vprašanje: Katero maksimalno vrednost lahko zavzame r, da bi dobili še enako stabilno oscilacijo (n.pr. s periodo 2)? Če izberemo $2<r<\sqrt{6}$, dobimo oscilacije s periodo 2, če pa izberemo r malo večji od $\sqrt{6}$, dobimo oscilacije s periodo 4 itd (slika 2.5).



Slika 2.5 Verhulstov dinamični proces

Vprašali se boste v kakšni povezavi je vse to skupaj s fraktali? Verhulstov proces je enodimenzionalen, Mandelbrot pa je raziskal popolnoma enak problem, le v 2D prostoru. Odločil se je opazovati kompleksna števila namesto realnih, pri čemer je sledil procesu $x_0, x_1, x_2, \ldots$, v ravnini, raje kakor na premici.

Dejal je, da imamo v ravnini **stekališča** (attractors), ki se "borijo" za svoj vpliv na ravnini. Točka x_0 se v dinamičnem procesu približuje enemu ali drugemu stekališču, lahko pa je na meji (ločnici) med dvema stekališčema in se ne more opredeliti za enega. Mandelbrotov proces je matematično ekvivalenten Verhulstovem procesu. Izhaja iz preproste formule:

$$x_{n+1} = f(x_n) = x_n^2 + c$$

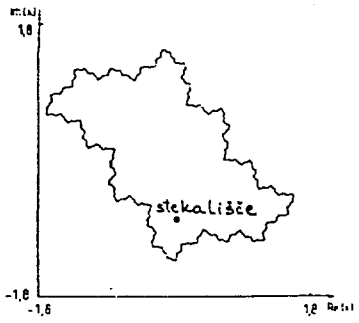
pri kateri ločimo dve situaciji:

1. Fiksiramo vrednost c in dovolimo, da x_0 zavzame različne vrednosti kompleksnih števil.
2. Fiksiramo x_0 in dovolimo spremembo c.

Pri prvi situaciji lahko postavimo c=0 ali c≠0.
a) Če je c=0 dobimo zaporedje iteracij $x_0, x_0^2, x_0^4, \ldots$, pri katerem ločimo tri različna stanja glede na vrednost x_0:

- Števila zaporedja postajajo vse manjša in manjša. Takšna situacija nastane, kadar je $|x_0| < 1$. Zaporedje limitira proti nič in pravimo, da je nič stekališče procesa $x \Rightarrow x^2$.
- Števila zaporedja postajajo vse večja in večja. Do tega primera pridemo, kadar je $|x_0| > 1$. Zaporedje limitira proti neskončnosti in pravimo, da je neskončnost stekališče procesa $x \Rightarrow x^2$.
- $|x_0| = 1$. V tem primeru so števila zaporedja na meji med dvema stekališčema. Mejo sestavlja krožnica enotskega kroga, ki razdeli ravnino na dve regiji. Nič predstavlja notranje stekališče, neskončnost pa zunanje stekališče.

b) Če predvidimo $c \neq 0$ dobimo zaporedje iteracij $x_0, x_1, x_2, \ldots$, pri katerem lahko prav tako nastanejo prej naštete možnosti. Razlika je le v tem, da notranje stekališče nima več vrednosti 0, temveč drugačno vrednost, in da meja ni več lepo ukrivljena, temveč je "nazobčana". Prav s tem pa smo porušili pravilnost geometrijskih likov in dobili lik, ki je podoben naravnemu. Imenujemo ga *fraktal* (slika 2.6).



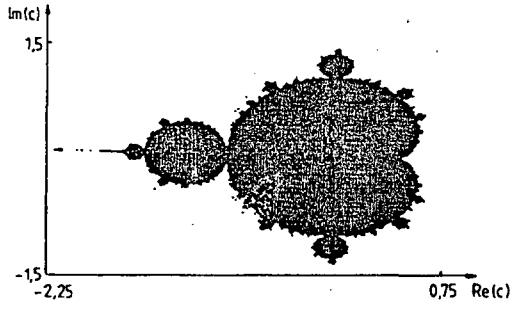
Slika 2.6 Fraktal

Pri podrobnejši analizi meje ugotovimo dve zanimivi lastnosti :

- Če vzamemo povečevalno steklo in pogledamo delček meje od blizu, ugotovimo, da je popolnoma enak, kakor če ga pogledamo brez povečevalnega stekla. To lastnost imenujemo *samopodobnost*.
- druga lastnost je *zrcalnost* .Če pogledamo obliko meje v enem kotu, ugotovimo njeno zrcalno sliko v nasprotnem kotu.

Fraktale, ki posedujejo takšne lastnosti, imenujemo **Julilove množice** (po francoskem matematiku Gaston Julii) [7].

Druga možnost je, da fiksiramo $x_0 = 0$ in dovolimo spremembo c po vsej kompleksni ravnini. Na tak način dobimo novo množico, ki jo imenujemo **Mandelbrotova množica M** (B. Mandelbrot je prvi objavil njen opis in jo prikazal na računalniku, slika 2.7).

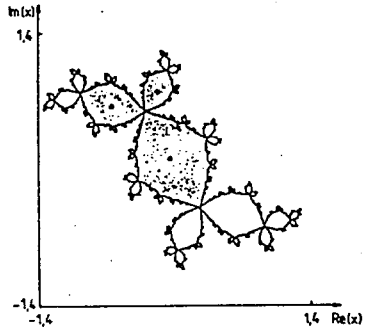


Slika 2.7 Mandelbrotova množica pri $x_0 = 0$

Zanimive so naslednje ugotovitve:

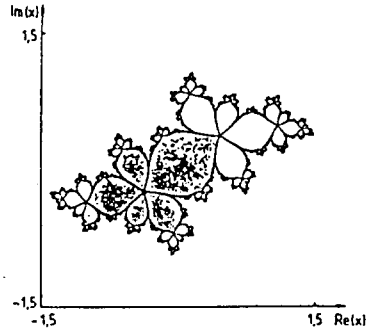
- Če izberemo c iz notranjosti glavnega dela telesa Mandelbrotove množice in ga po prvem postopku razvijemo v Juliovo množico, dobi Juliova množica obliko deformirane, nazobčane krožnice, ki obkroža stekališče (slika 2.6, podoben primeru 1 na sliki 2.13).

- Če izberemo c v notranjosti enega izmed "popkov" Mandelbrotove množice in ga razvijemo v Juliovo množico, bo Juliovo množico sestavljalo več deformiranih, nazobčanih krožnic, kjer vsaka izmed njih obkroža svoje stekališče (slika 2.8, primer 3 na sliki 2.13).



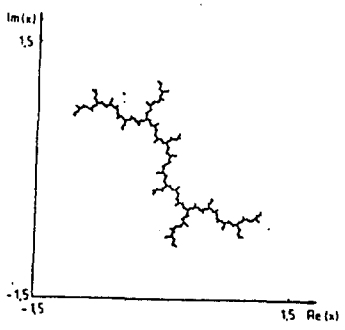
Slika 2.8 Primer fraktala, ki se nahaja v notranjosti enega izmed "popkov" Mandelbrotove množice

- Če izberemo c na meji, kjer se "popek" dotika glavnega dela telesa Mandelbrotove množice in ga razvijemo v Juliovo množico, dobimo t.i. parabolični primer. Zanj je značilno, da se več nazobčanih krožnic steka v eno točko (slika 2.9, primer 8 s slike 2.13).



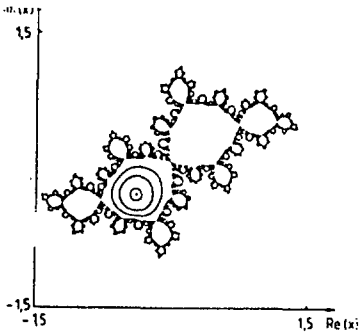
Slika 2.9 Parabolični primer fraktala

- Če izberemo c na "anteni" Mandelbrotove množice in ga razvijemo v Juliovo množico, dobimo fraktal imenovan *dendrit*. Zanj je značilno, da ima eno samo stekališče, ki je v neskončnosti. Zaradi tega tak fraktal nima notranjosti (slika 2.10, primer 4 na sliki 2.13).



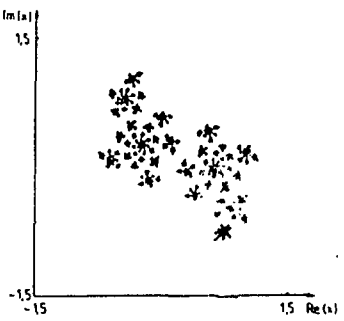
Slika 2.10 Primer dendrita

- Če izberemo c kjerkoli drugje na meji Mandelbrotove množice in ga razvijemo v Juliovo množico, pride do pojavov Siegelovih diskov. Če točka v iteracijskem postopku doseže Siegelov disk, bo v njem tudi ostala. To pomeni, da bo v krogu rotirala okoli stekališča, samega stekališča pa ne bo nikdar dosegla (slika 2.11, primer 9 s slike 2.13).



Slika 2.11 Primer fraktala s Siegelovimi diski

- Če izberemo c izven Mandelbrotove množice in ga razvijemo v Juliovo množico, bo Juliova množica nepovezana. To pomeni, da razpade v množico točk in tak fraktal imenujemo *Fatouov prah* (slika 2.12, primer 11 na sliki 2.13).



Slika 2.12 Fatouov prah

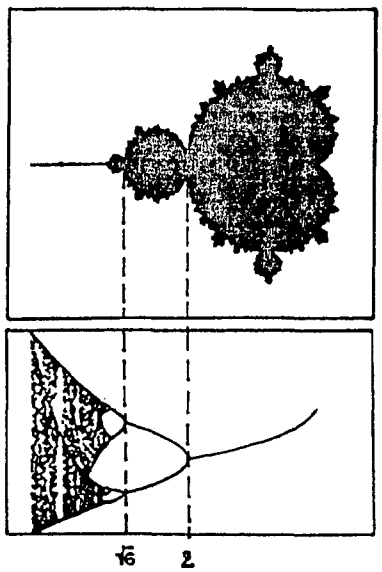
1983. leta je Dennis Sullivan dokazal, da so to edine oblike, ki jih Juliova množica lahko zavzame. Obstaja še ena možnost, tako imenovani Hermanov obroč (Herman ring), ki pa ne nastane v primeru $x \mapsto x^2$. Teoretično je dokazano, da lahko do njegovega pojava pride v drugih primerih, čeprav praktično še ni bil nikdar opazovan [7].

Slika 2.13 prikazuje nekaj Juliovih množic, ki jih dobimo z ustrezno izbiro parametra c iz Mandelbrotove množice.

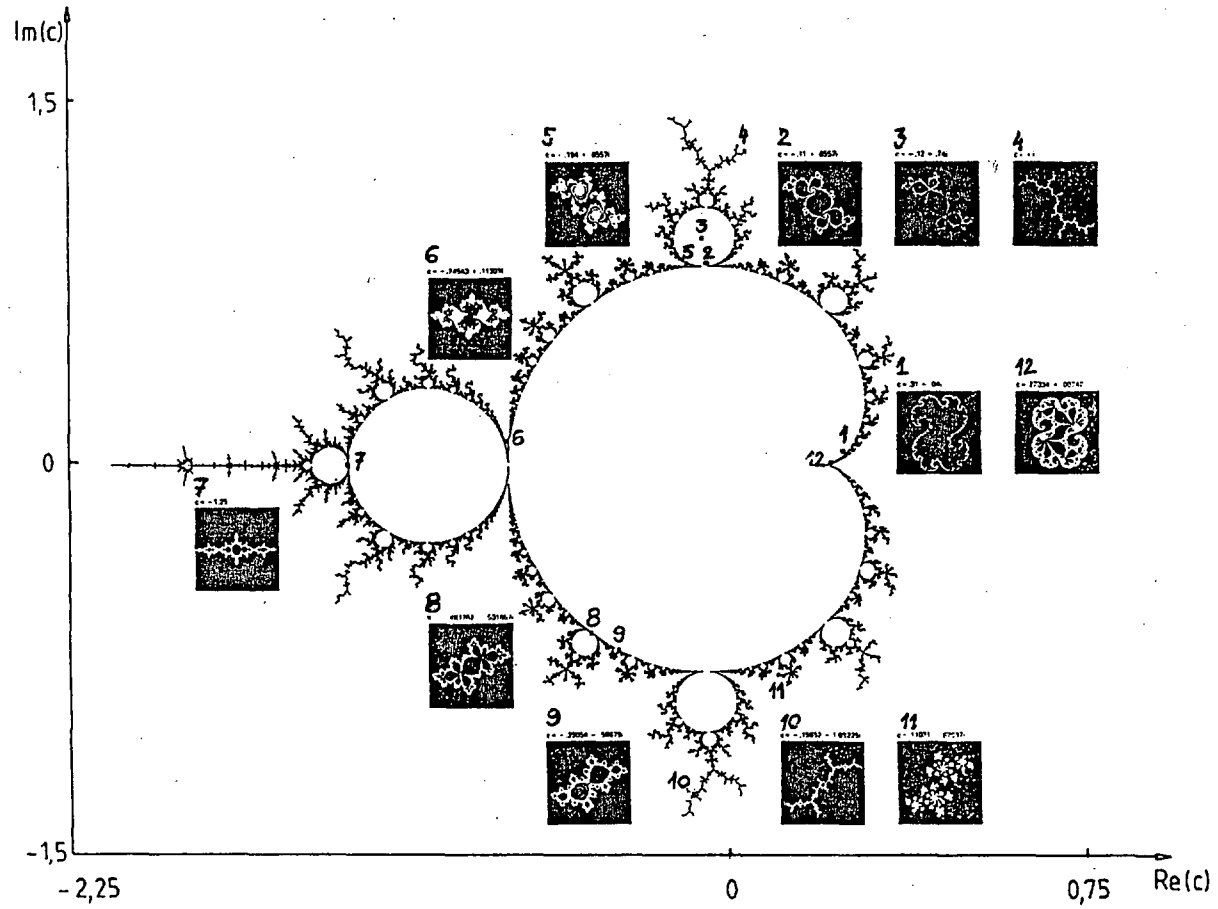
2.1 Podobnosti Mandelbrotove množice z Verhulstovim dinamičnim procesom

Raziskave so pokazale, da ima Mandelbrotova množica veliko skupnih točk z Verhulstovim dinamičnim procesom (slika 2.14). Pri Mandelbrotovi množici ugotovimo stabilno področje ter oscilacije s periodo $2, 4, 8, 16, \ldots$, ravno tako kakor pri Verhulstovem dinamičnem procesu.

Stabilno področje zajema območje med $c=0.25$ in $c=-0.75$, oscilacije s periodo 2 se pojavljajo v krogu s polmerom 1/4 in središčem v točki $c=-1$, oscilacije s periodo 4 pa okoli središča $c=-1.3107$ itn. Točke, kjer nastajajo "popki" na realni osi na sliki Mandelbrotove množice, ustrezajo točkam, kjer prihaja do podvajanj periode v Verhulstovem procesu.



Slika 2.14 Podobnosti Mandelbrotove množice z Verhulstovim dinamičnim procesom

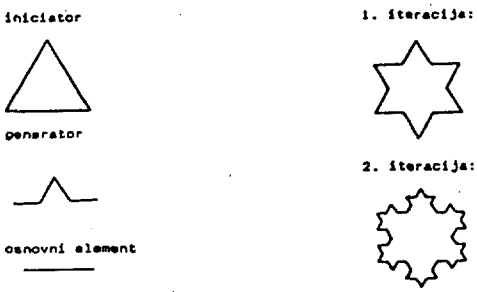


Slika 2.13 Juliove množice kot sestavni del Mandelbrotove množice

3. GEOMETRIČNI FRAKTALI

Geometrični fraktali se od naravnih razlikujejo po tem, da ustvarjajo slike, ki jih je mogoče vnaprej predvideti. Primer geometričnega fraktala je Kochova snežinka (slika 3.1), ki jo lahko generiramo s pomočjo naslednjega algoritma:

```
Kochova_krivulja := iniciator;
loop
    vsak osnovi element v Kochovi_krivulji
    zamenjaj z generatorjem
endloop.
```

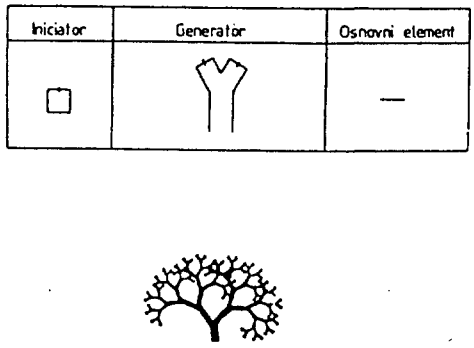


Slika 3.1 Primer geometričnega fraktala-Kochova snežinka

Opazimo lahko, da s ponavljanjem iteracij v globino ustvarjamo čedalje bolj natančno sliko. S tem dobimo občutek, da je objekt (snežinka), ki ga opazujemo, zelo blizu. Pri majhnem številu iteracij dobimo občutek oddaljenosti objekta.

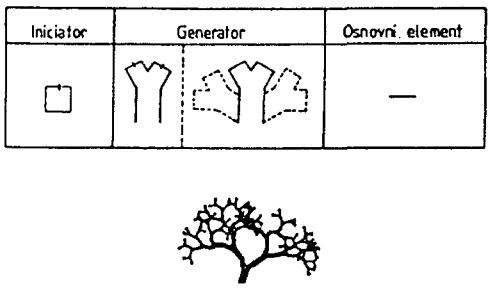
Kochova snežinka predstavlja najprimitivnejši fraktal v verigi geometričnih fraktalov. To pomeni, da lahko v osnovni algoritem uvedemo dodatna pravila in na tak način dobimo zanimivejše in bolj kompliciran fraktale.

Pravilo 1: Nekatere daljice iniciatorja zaznamujemo kot koristne in jih v iteracijskem postopku spreminjamo (slika 3.2).



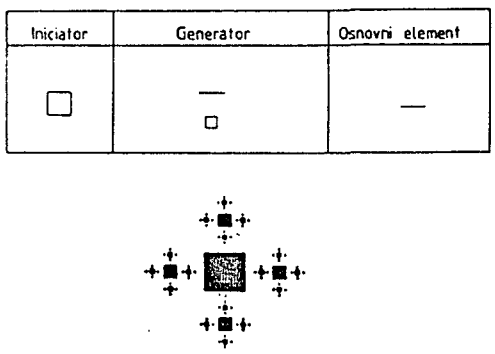
Slika 3.2 Drevo, narisano s pomočjo pravila 1 in zapolnitvijo notranjosti

Pravilo 2: Dodamo funkcijo, ki bo dovoljevala generatorju naključni premik na levo ali desno (slika 3.3).



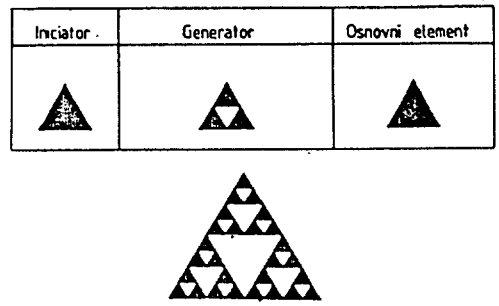
Slika 3.3 Drevo, narisano s pomočjo pravila 2 in zapolnitvijo notranjosti

Pravilo 3: Generator sestavimo iz dveh ali večih delov (slika 3.4).



Slika 3.4 Vzorec, narisan s pravilom 3 in zapolnitvijo kvadratkov

Pravilo 4: Predpostavimo, da je osnovni element enak iniciatorju (slika 3.5).



Slika 3.5 Trikotnik Sierpinskega, narisan s pomočjo pravila 4

Seveda lahko ta pravila združimo in jih uporabimo skupaj. Na tak način porušimo geometrijsko natančnost in dobimo nepravilne krivulje, ki imajo obliko oblakov, gorovja itn.

4. UPORABA METODE IFS ZA GENERIRANJE GEOMETRIČNIH FRAKTALOV

Trikotnik Sierpinskega iz pravila 4 je mogoče narisati tudi na na drugačen način in sicer tako, da izberemo začetno točko $\vec{p}_{0,h} = [0,0,1]$ (dano s homogenimi koordinatami) ter eno izmed treh transformacij:

$$T_1 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ T_1 je skalirna matrika.

$$T_2 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$ T_2 je produkt skalirne in translacijske matrike.

$$T_3 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$ T_3 je produkt skalirne in translacijske matrike.

Transformacijo izberemo naključno, jo izvršimo nad začetno točko in dobimo točko $\vec{p}_{1,h}$. Nato naključno izberemo naslednjo transformacijo in jo izvršimo nad točko, dobljeno v prejšnjem koraku. Verjetnosti za izbiro posamične transformacije so enake. Postopek nadaljujemo in po določenem številu iteracij pridemo do trikotnika Sierpinskega.

Na prvi pogled se zdi čudno kako lahko naključna izbira ene od treh transformacij privede v iterativnem postopku do tako složne harmonije, kakor je trikotnik Sierpinskega. Področje v matematiki, ki se ukvarja s takšnimi transformacijami, je teorija iterativnih funkcijskih sistemov (IFS) in od tod tudi ime za našo metodo.

5. UPORABA FORMALNIH JEZIKOV NA PODROČJU FRAKTALNE TEORIJE

Nekatere objekte, na primer drevje in rastlinje, je mogoče predstaviti s pomočjo gramatik PRG (parallel rewriting grammars). Benoit Mandelbrot v svojih definicijah [6] ne dovoljuje , da bi objekte, dobljene na tak način, imenovali fraktali. Trdi namreč, da je pojem fraktala strogo geometrijski, medtem ko s pomočjo formalnih jezikov dobimo strukturirano zgradbo.

Objekte, nastale s pomočjo gramatik, imenujemo *graftali* [8]. Graftali se od fraktalov razlikujejo po tem, da ne ustvarjajo strogo zrcalnih in samopodobnih slik ter ne dovoljujejo nikakršne naključnosti pri uporabi produkcijskih pravil.

Značilnost gramatik PRG je zaporedna uporaba produkcijskih pravil (nad vsakim simbolom izvršimo produkcijsko pravilo), prav tako ne ločijo terminalnih simbolov od neterminalnih.

Oglejmo si primer!

Dana naj bo abeceda $\Sigma = \{0, 1, (,), [,]\}$

začetni simbol S=0 ter

produkcijska pravila:
$0 \rightarrow 1[0]1(0)0$
$1 \rightarrow 11$
$[\rightarrow [$
$] \rightarrow]$
$) \rightarrow)$
$(\rightarrow ($

Uporabimo naslednji algoritem:

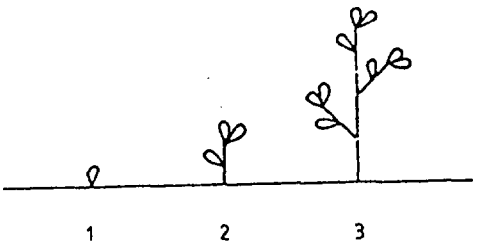
```
for i:= 1 to stevilo_generacij do
  nad vsakim simbolom v nizu uporabi produkcijsko
  pravilo
```

Pri stevilu_generacij = 3 dobimo:

stevilo_generacij	niz
1	0
2	1[0]1(0)0
3	11[1[0]1(0)0]11(1[0]1(0)0)1[0]1(0)0

Dobljeni niz želimo grafično predstaviti v obliki drevesa, zato uporabimo preprosto interpretacijo:

0...nariši list
1...nariši črto
[...začetek vejitve v levo
]...konec vejitve v levo
(...začetek vejitve v desno
)...konec vejitve v desno



Slika 5.1 Prikaz postopka, kako s pomočjo predhodno definirane gramatike tvorimo graftalno sliko. Vejitveni kot znaša 45 stopinj.

Seveda obstajajo še bolj kompliciranе interpretacije našega niza. Tako lahko npr. spreminjamo vejitveni kot glede na smer vetra, čvrstost, velikost drevesa itn. Prav tako lahko predpostavimo veje v obliki valja, katerega premer in višina sta odvisna od položaja enice v nizu. Ta dodatni pogoj omogoča, da v določenem trenutku dobimo manjše in tanjše veje.

6. OPIS ALGORITHOV ZA GENERACIJO FRAKTALOV

Pri oblikovanju 2D fraktalnih slik smo upoštevali dinamični proces nad polinomom druge stopnje $z \Rightarrow z^2 + c$ (namesto x pišimo z). Kompleksni števili z in c smo razdelili na realni in imaginarni del (z=x+iy, c=p+iq) tako, da je dinamični zakon prešel v obliko:

$$x_{n+1} = f(x_n, y_n, p) = x_n^2 - y_n^2 + p$$
$$y_{n+1} = g(x_n, y_n, q) = 2x_n\, y_n + q$$

Kot že omenjeno, smo do oblikovanja slik prišli po dveh poteh:

1. Upoštevali smo ravnino xy in fiksni vrednosti p ter q. Za vsako točko ravnine smo ugotovili njeno dinamično povezanost do ustreznega stekališča. Na tak način smo raziskali strukturo Juliovih množic.
2. Upoštevali smo ravnino pq ter fiksno točko (x,y). Tak postopek je produciral slike Mandelbrotove množice.

Program za izračun dinamičnega procesa ter določitev barv je tekel na VAX 8800. Rezultate je shranjeval na datoteke, ki smo jih kasneje prenesli na PC-AT. Tam je tekel program za izris fraktalnih slik. Napisan je bil v turbo pascalu.

Pri izrisu slik smo uporabili 16 barv. Povedale so, kako dolgo je potrebovala točka (x,y) da se je približala stekališču. Če se točka po določenem številu iteracij (5000) ni približala stekališču, smo jo pobarvali s črno barvo. Velikost okna (a x b) smo izbrali (511 x 350) pikslov.

Algoritma 1 in 2 smo uporabili za generiranje naključnih fraktalov. S pomočjo algoritma 3, pa smo

narisali geometrična fraktala trikotnik Sierpinskega in list praproti.

Algoritem 1: Juliova množica

korak 0: Izberi parameter c=p+iq
Določi velikost okna: $x_{min} = y_{min}$
$x_{max} = y_{max}$
$\Delta x=(x_{max} - x_{min})/(a-1)$
$\Delta y=(y_{max} - y_{min})/(b-1)$
Za vse točke zaslona (n_x,n_y), $n_x = 0,1,\ldots,a-1$, $n_y = 0,1,\ldots,b-1$, izvrši naslednje korake.

korak 1: $x_0 = x_{min} + n_x\Delta x$
$y_0 = y_{min} + n_y\Delta y$
števec_iteracij = 0

korak 2: $x_{n+1} = x_n^2 - y_n^2 + p$
$y_{n+1} = 2x_n y_n + q$
Povečaj števec_iteracij za 1.

korak 3: Izračunaj $r^2 = x_n^2 + y_n^2$
(i) if r>2 then izberi barvo in goto korak 4
(ii) if števec_iteracij = 5000 then izberi barvo 0 (črno) in goto korak 4
(iii) if r ≤ 2 and števec_iteracij<5000 then goto korak 2

korak 4: Pobarvaj točko (n_x,n_y) z ustrezno barvo, izberi naslednjo točko in pojdi na korak 1.

Da bi izboljšali čas računanja, smo upoštevali, da točki (x,y) in (-x,-y) producirata enak rezultat, saj je slika simetrična glede na izhodišče. Pozorni moramo biti tudi na izbiro koordinat okna, saj ob njihovi nepravilni izbiri lahko dobimo nezanimivo sliko.

Algoritem 2: Mandelbrotova množica

korak 0: Izberi p_{min}, p_{max}, q_{min}, q_{max}.
$\Delta p = (p_{max} - p_{min})/(a-1)$
$\Delta q = (q_{max} - q_{min})/(b-1)$
Za vse točke zaslona (n_p,n_q), $n_p = 0,1,\ldots,a-1$, $n_q = 0,1,\ldots,b-1$, izvrši naslednje korake.

korak 1: $p_0 = p_{min} + n_p\Delta p$
$q_0 = q_{min} + n_q\Delta q$
števec_iteracij = 0
$x_0 = y_0 = 0$

korak 2: $x_{n+1} = x_n^2 - y_n^2 + p$
$y_{n+1} = 2x_n y_n + q$
Povečaj števec_iteracij.

korak 3: Izračunaj $r^2 = x_n^2 + y_n^2$
(i) if r>2 then izberi barvo goto korak 4
(ii) if števec_iteracij=5000 then izberi barvo 0 (črno) in goto korak 4
(iii) if r ≤ 2 and števec_iteracij<5000 then goto korak 2

korak 4: Pobarvaj točko (n_p,n_q) z ustrezno barvo, izberi naslednjo točko in pojdi na korak 1.

Pri tretjem algoritmu smo začetno točko (x_0,y_0) izbrali (0,0). Število iteracij je bilo 32000. Program je v celoti tekel na PC.

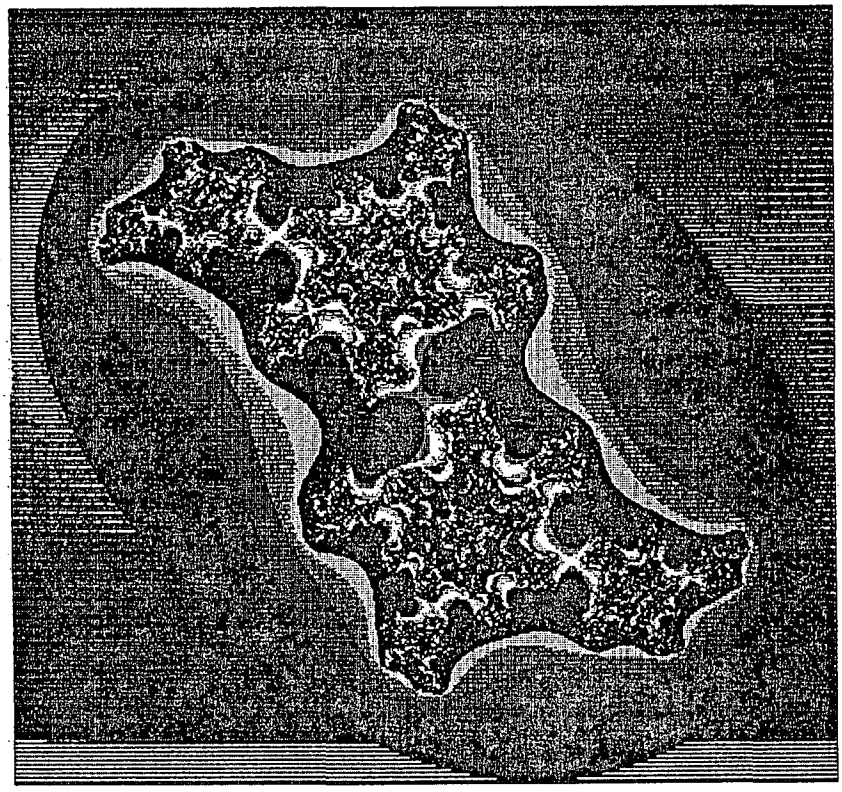
Algoritem 3 : metoda IFS

korak 0: Okno opazovanja V naj bo velikosti X × Y, pri čemer je X × Y resolucija zaslona.
Okno V razdelimo na L × M kvadratov (označimo jih z V_{ij}) velikosti dolž (L=L/dolž, M=M/dolž).
Izberi število_iteracij ≥ L × M.
Polje V inicializiraj na 0.
Izberi začetno točko $(x_0,y_0) \in R^2$.
Število_barv=16.

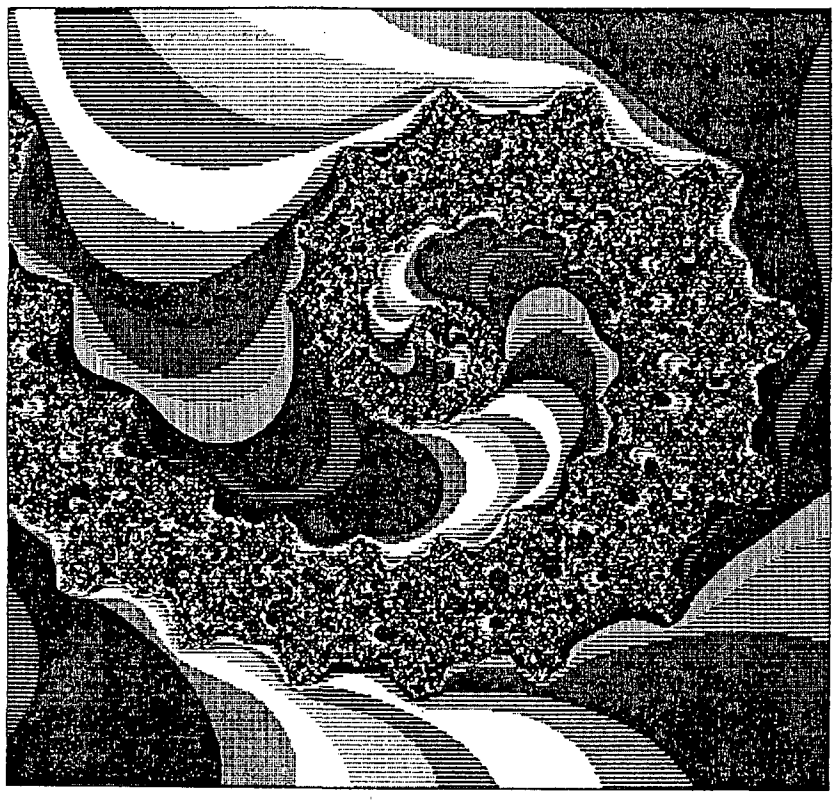
korak 1:
```
for n = 0 to število_iteracij do
begin
   (* izberi nakjučno transformacijo M_k *)
   rand = random število med [0,1];
   verjetnost = p_1;
   k = 1;
   while (verjetnost < rand) do
   begin
      k = k+1;
      verjetnost = verjetnost + p_k;
   end;
   (* izvrši transformacijo nad točko *)
   [ x_{n+1} y_{n+1} 1 ] = [ x_n y_n 1 ] M_k
   n_1 = int(x_{n+1});
   n_2 = int(y_{n+1});
   (* zaznaj "obisk" v kvadratu V_ij *)
   V[n_1][n_2] = V[n_1][n_2] + 1;
end;
```

korak 2: Poišči maksimalno vrednost max v polju V.
Določi barvo pravokotnika V_{ij}:
barva_pravokotnika=barva(V[i][j]*15/max)
in ga pobarvaj.

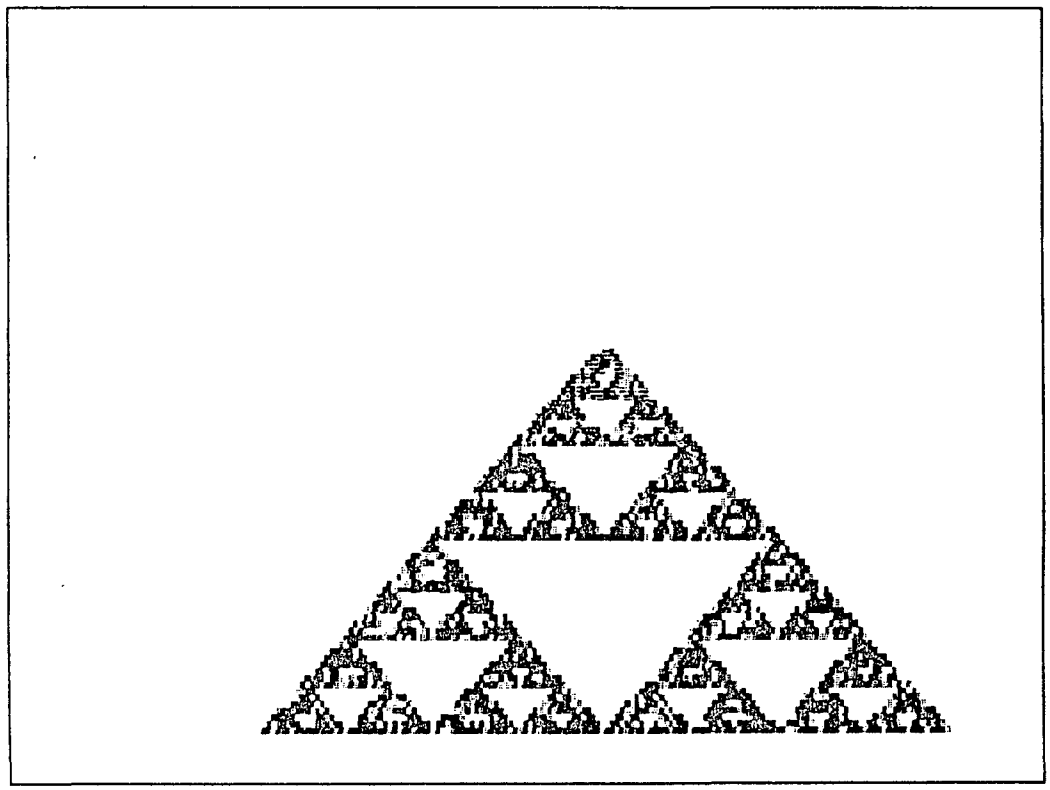
Slike 6.1, 6.2 in 6.3 so bile izrisane na barvnem brizgalnem risalniku Tektronix 4696.



Slika 6.1 Juliova množica pri c = 0.11031 - 0.670371



Slika 6.2 Mandelbrotova množica



Slika 6.3 Trikotnik Sierpinskega dobljen s pomočjo metode IFS

7. ZAKLJUČEK

Znanost in umetnost sta dva nasprotujoča si načina za izražanje naravnega sveta - prvi analitičen, drug intuitativen. S pomočjo fraktalov smo ti dve poti združili in dokazali, da sta odvisni druga od druge.

Fraktali so zanimivi zaradi visoke stopnje vizualne kompleksnosti, čeprav so v svojem bistvu izredno preprosti. Matematična osnova fraktalne teorije je enostavna, saj temelji na ponavljanju. Prav zaradi tega jo je bilo izredno lahko uporabiti na področju računalništva. Algoritmi za kreiranje fraktalnih slik potrebujejo malo podatkov, s pomočjo rekurzije pa so zelo uspešni.

Teorija fraktalov je pomagala zlomiti strogo geometrijo likov. Trendi v svetu pa segajo še dalje. Področje uporabe fraktalov se naglo širi - od biologije do tekstilne industrije pa vse do drugih, predvsem barvnih grafičnih izdelkov. Možnosti je torej ogromno - ta članek opisuje le delček njih.

8. LITERATURA

[1] M.Barnsley: "Harnessing Chaos for Image Synthesis", Computer Graphics, Volume 22, Number 4, avgust 1988

[2] P.Blanchard: "Complex Analytic Dynamics on the Riemann Sphere", Amer.Math Soc. 11, 1984, str.85-141

[3] S.Demko: "Construction of Fractal Objects with Iterated Function Systems", Computer Graphics, Volume 19, Number 3, 1985, str.271-278

[4] M.Gervautz: "Fractals in Computer Graphics", Proceeding of The Second Austro-Yugoslav Conference on Computer Graphics, str.96-103

[5] J.A.Kaandorp : "Interactive Generation of Fractal Objects", Eurographics'87, str.181-196

[6] B.B.Mandelbrot: "The Fractal Geometry of Nature", W.H.Freeman, San Francisco, 1982

[7] H.O.Peitgen, P.H.Richter: "The Beauty of Fractals", Springer Verlag, Berlin, Heidelberg, New York, 1986

[8] A.R.Smith: "Plants, Fractals and Formal Languages", Computer Graphics, Volume 18, Number 3, julij 1984, str.1-10

[9] A.Terčelj: "Fraktali - grafične skrivnosti računalniških umetnikov", Informatica 3/87, str.46-50

PRIMENA METODA INŽENJERSTVA ZNANJA U OBRAZOVANJU

INFORMATICA 4/89

Keywords: method, knowledge engineering, artificial intelligence, education

Ljubomir Jerinić, Zoran Budimac, Đura Paunić i Mirjana Ivanović

U radu su opisane metode inženjerstva znanja kao podoblasti veštačke inteligencije, i mogućnosti primene tih metoda u obrazovanju. Kao najpogodnija metoda reprezentacije znanja za potrebe obrazovanja izabran je sistem okvira Marvina Minskog. Na osnovu takvog pristupa razvijen je programski sistem OSOF za prikupljanje, internu reprezentaciju i korišćenje znanja u svim vidovima i nivoima obrazovanja.

1. UVOD

Prema [1], veštačka inteligencija (VI) je deo računarskih nauka u kojima se projektuju inteligentni računarski sistemi, koji se ponašaju na način sličan inteligenciji u ljudskom ponašanju, za razumevanje prirodnih jezika, učenje, rezonovanje, rešavanje problema i sl. VI je interdisciplinarna nauka i potekla je iz istraživanja iz domena simboličke obrade podataka i dela psihologije o rasuđivanju. VI istražuje simboličku obradu i heurističke procese zaključivanja i rezonovanja, kao i predstavljanje znanja u obliku pogodnom za zaključivanje uz pomoć računara [2].

Osnovni pravci istraživanja u okviru VI su: razumevanje prirodnog govora, mašinsko učenje, automatsko programiranje, računarska vizija, inteligentna robotika, inženjerstvo znanja itd.

Razvojem računarske tehnologije i metoda VI u zadnjih deset godina, približavanjem računara svim uzrastima, kao i uvođenjem računara u škole, istraživačima VI obrazovanje postaje interesantno polje rada. Deo VI, inženjerstvo znanja, po svojoj definiciji, metodama i rezultatima postaje direktno primenljivo i u obrazovanju, sa ciljem krajnje individualizacije obrazovnog procesa. Davnašnja težnja da jedan nastavnik ili profesor obrazuje jednog učenika ovakvim pristupom postaje realnost, kao i napredovanje svakog učenika prema njegovim sposobnostima.

U radu se dalje opisuju metode inženjerstva znanja i njihova primena u obrazovanju. Data je definicija i klasifikacija načina pretstavljanja znanja, prihvaćena u inženjerstvu znanja. Iskorišćena je metoda M. Minskog [6] za realizaciju univerzalnog programskog paketa OSOF [3] za primenu računara u obrazovanju.

2. INŽENJERSTVO ZNANJA

U ranom razdoblju istraživanja u ovoj podoblasti VI, do sredine 70-tih godina, težilo se (pod uticajem psihologije), iznalaženju opštih metoda rešavanja problema "ekspertize" znanja, zasnovanih na opštim principima zaključivanja sa psihološkog aspekta. Ovakav pristup se pokazao neefikasnim za Von Neuman-sku organizaciju računara i sa malom primenljivošću u praksi. Nedostatak je prevashodno što se unutar opšteg generalizovalo specifično znanje relativno disjunktnih oblasti.

Krajem sedamdesetih godina se sa paradigme zasnovane na zaključivanju prešlo na novu paradigmu zasnovanu na znanju. Oblast delovanja istraživača inženjerstva znanja postaje istovetna sa oblastima delovanja stručnjaka iz pojedinih uskih oblasti: prikupljanje specifičnih znanja i iskustava, te potom i njihova primena u rešavanju određene grupe problema. Preduslov za ovakvo, heurističko, rešavanje problema je izbor pogodne reprezentacije relevantnog znanja kome inteligentni program može lako da pristupi, dok mehanizam zaključivanja, tj. mehanizam korišćenja tako memorisanog znanja treba da je jednostavan i zasnovan na tom znanju umesto na opštim principima ili nekakvim funkcijama komplikovanim za izračunavanje.

Ljudsko znanje se kodira određenim metodama u module koji se u mehanizmu zaključivanja aktiviraju uzorcima: "sirovi" podaci, "obrađeni" podaci, parcijalna rešenja, neočekivane situacije, greške i sl. Ovakvi uzoračko vođeni moduli imaju niz prednosti u odnosu na nekakav opšti algoritam zaključivanja:

-predstavljanje znanja u delovima je primereniji načinu memorisanja znanja eksperata,

-programiranje sa ovakvim modulima omogućava razvoje inteligentnih sistema u koracima, programi se lako modifikuju i proširuju, a greške unutar znanja se popravljaju bez izmene koda programa, i sl.

Proces konstruisanja jednog inteligentnog sistema obuhvata sledećih pet oblasti:

-prikupljanje i sistematizacija relevantnog znanja: od eksperta ili mašinskim učenjem na primerima,

-reprezentacija znanja: izbor pogodnog načina kojim se velika količina znanja može predstaviti pomoću simboličkih struktura podataka unutar računara, pogodnih za zaključivanje. Odabrana struktura treba da omogući fleksibilne izmene i dopune memorisanog znanja,

-primena znanja: planiranje i kontrola rešavanja problema, heurističko zaključivanje, tačnost i efikasnost rada inteligentnih procesa, koje memorisano znanje predstavlja,

-generisanje objašnjenja: interakcija računar-čovek, objašnjenje zašto je problem rešen na jedan a ne na neki drugi način, mogućnost učenja na greškama i uticanja čoveka na proces zaključivanja,

-obrazovanje: smešteno znanje se uz izvesne ograde može koristiti i za stvaranje

inteligentnih sistema učenja.

3. INŽENJERSTVO ZNANJA I OBRAZOVANJE

Računar u obrazovanju sa stanovišta nastave se može posmatrati kao novo nastavno sredstvo i kao upravljač nastavnog procesa. Za razliku od konvencionalnih nastavnih sredstava (grafoskop, diaskop, interna televizija, responderi i dr.) računar u nastavni proces unosi značajnu novinu - mogućnost obostrane komunikacije računar - učenik. Nijedno od konvencionalnih nastavnih sredstava ne može da odgovara na pitanja učenika i da na taj način usmerava nastavni proces. Kao upravljač nastavnog procesa, računar u obrazovanju se posmatra kao neposredni izvršioc nastavnog procesa kreiranog putem programske podrške i upravljač raznih pomoćnih uređaja nastavnog procesa (laboratorijski uređaji, responderi i dr.).

Ovako posmatrana primena računara u obrazovnom procesu otvara mogućnost primena metoda i tehnika inženjerstva znanja u obrazovanju i otvara novo polje istraživanja - projektovanje inteligentnih sistema učenja. Projektovanje ovakvih sistema učenja zahteva interakciju metoda inženjerstva znanja sa jedne strane i metoda pedagogije, metodike, didaktike i psihologije sa druge.

Pretpostavka da se znanje koje sadrže ekspertni sistemi standardnog tipa može iskoristiti i u svrhe obučavanja, demantuje se u praksi jer su takvi inteligentni programi loši učitelji [4]. Uzrok je upravo izostavljanje osnovnih principa pedagogije i metodike pri kreiranju programskih sistema čija je svrha prvenstveno konsultanska pomoć pri rešavanju problema.

Kreiranje posebnih inteligentnih sistema namenjenih obrazovanju, mora rešiti sledeće probleme:

-kako predstaviti znanje koje učenik treba da usvoji,

-kako opisati pojmove koji se usvajaju,

-kako usaglasiti mehanizam korišćenja takvog znanja sa metodičkim i pedagoškim principima usvajanja znanja,

-kako iskoristiti dobre osobine svih vrsta konvencionalnih načina prenošenja i usvajanja znanja: predavanje, testiranje, eksperiment, opažanje, programirana nastava i dr.

4. SISTEMI REPREZENTOVANJA ZNANJA

Znanje se može definisati kao simbolička reprezentacija činjenica i relacija među njima. Reprezentacija znanja je način predstavljanja znanja, kao i način kako se ono može povezivati sa drugim znanjem, koristiti za rešavanje novih situacija i izvoditi novo znanje.

U [8] reprezentacija znanja se definiše kao skup sintaksnih i semantičkih konvencija koje omogućavaju opisivanje stvari. Sintaksa reprezentacije je skup pravila za formiranje, kombinovanje i uređenje izraza u jeziku reprezentacije. Semantika reprezentacije određuje kako se sintaksno valjani izraz koji reprezentuje znanje interpretira, tj. kako se iz date forme može izvesti i koristiti znanje.

Reprezentacija znanja je bitan element inženjerstva znanja koji omogućava sistematičan način kodiranja znanja stručnjaka o nekom problemu unutar neke šire kategorije. Ona implicitno obuhvata i organizaciju podataka u računaru za memorisanje znanja. Postoji niz notacijskih sistema za predstavljanje znanja u računaru. Zajednička karakteristika svih je da treba da obezbede brz pristup znanju i da se znanje može lako koristiti pomoću manje-više prirodnih mehanizama zaključivanja ili izvođenja. Programeru nakon odabiranja notacijskog sistema ostaje da izabere najpogodniju strukturu podataka koja se implementira zadržavajući sve dobre osobine izabranog notacijskog sistema.

Važni kriterijumi za izbor notacijskog sistema i organizaciju podataka za reprezentaciju znanja su:

-logička dovoljnost tj. da formalizam može na pogodan način da opiše znanje,

-efikasnost pristupa pri obradi znanja,

-iskoristivost, tj. reprezentovano znanje se može upotrebiti u rešavanju problema za koje su ta znanja dovoljna.

Za reprezentaciju znanja u primeni inženjerstva znanja se koriste sledeći formalizmi: produkcijska pravila, strukturni objekti i predikatska logika.

Svi navedeni formalizmi se implementiraju sami ili u nekoj kombinaciji, u uzoračko vođenim sistemima. Struktura podataka koja pretstavlja unutrašnju reprezentaciju izabranih formalizama pretstavljanja znanja je zavisna od problema koji se rešava, sistema zaključivanja, interpretatora mehanizma procesiranja znanja, računara na kome se implementira i programskog jezika izabranog za realizaciju određenog inteligentnog sistema.

Sistemi koji koriste neki od navedenih formalizama, se sastoje od niza relativno nezavisnih modula koji omogućavaju prikupljanje i memorisanje znanja, formiranje odgovarajuće unutrašnje reprezentacije znanja, korišćenje prikupljenog znanja i sl.

Produkcijska pravila su formalizam za predstavljanje znanja koji se koristi i u teoriji automata, formalnim jezicima i dizajniranju programskih jezika. Sastoje se od skupa pravila oblika:

$$\text{if } (P_1 \wedge \ldots \wedge P_n) \text{ then } (A_1 \wedge \ldots \wedge A_m),$$

gde su P_i, i=1,...,n uslovi, a A_j, j=1,...,m zaključci.Uslovi su trojke objekt - atribut - vrednost oblika:

(Bor je_rudnik bakar)

dok su zaključci akcije koje se preduzimaju ako su uslovi zadovoljeni.

Korišćenjem ovog formalizma kodiraju se iskustvene veze (asocijacije) između uzoraka podataka i akcija koje sistem treba da preduzme kao posledicu zadovoljenosti uslova. Produkcijskim pravilima se kodiraju dva tipa znanja: opšte znanje relevantno za problem koji se opisuje tim znanjem i specifično znanje važno za posebne primene koje će koristiti to znanje. To specifično znanje se može i reprezentovati tradicionalnim tehnikama pretstavljanja znanja u bazama podataka. Neodređeno znanje koje, se takođe može koristiti je pretstavljeno trojkom:

$$\text{if } (P_1 \wedge \ldots \wedge P_n) \text{ then}$$

$$(A_1 \wedge \ldots \wedge A_m) \text{ with } V(0.7)$$

gde je V(0.7) verovatnoća da $(P_1 \wedge \ldots \wedge P_n)$ izvodi $(A_1 \wedge \ldots \wedge A_m)$.

Predikatska logika, u čijoj osnovi je propozicioni i predikatski račun, razvojem logičkog programiranja i logičkih programskih jezika, se koristi i kao sistem predstavljanja znanja. Znanje se predstavlja pomoću formula, kao npr.:

Svako voli nekog <==> $(\forall x)\ (\exists y)$ (voli(x y))

Ovakav načina opisivanja znanja se koristi u

reprezentaciji opštih činjenica za razliku od produkcijskih pravila u kojima znanje predstavlja iskustvene činjenice.

Strukturni objekti su opšti izraz za bilo koju šemu reprezentacije znanja. Osnovni delovi strukturnih objekata su analogni čvorovima i vezama u teoriji grafova, otvorima ("slot") i markerima teorije okvira ("frame"), ili otvorima i puniocima ("filler") konvencionalnih struktura podataka tipa slog. Oni obuhvataju sledeće formalizme: semantičke (asocijativne mreže), okvire, objektno orjentisane sisteme, strukture konceptualne zavisnosti ili skripte.

Predikatska logika i produkcijski sistemi kao formalizmi predstavljanja znanja se koriste za reprezentovanje različitih aspekata okoline. Međutim, u opštem slučaju, oni ne dozvoljavaju da struktuiramo znanje o toj okolini. Strukturni objekti uključuju mogućnost reprezentovanja strukture znanja, grupisanjem delova znanja u celine i relacija među znanjem, kao i pokazivača na akciju koja se preduzima korišćenjem tog znanja. Takođe se unutar strukturnih objekata predviđa rad sa podrazumevanim (default) vrednostima, otkrivanje grešaka kao i rad sa nepotpunim znanjem. Sve ove nabrojane osobine su direktno primenljive u predstavljanju znanja za obrazovne svrhe.

5. PREDSTAVLJANJE ZNANJA U SISTEMU OSOF

Znanje za potrebe obrazovanja se može organizovati u nastavne sekvence (lekcije) koje se sastoje od skupa međusobno povezanih članaka (pojmova). Članak sadrži minimalnu količinu zaokruženog znanja koga ucenik treba da savlada u jedinici vremena.

Svaki članak se sastoji od naziva, teksta kojim je on opisan, grafičke ilustracije ili simulacije i proizvoljnog broja zadataka kojim se može proveriti i utvrditi znanje opisano u članku.

Zadatak je opisan tekstom, slikom ili simulacijom, a odgovor se može eksplicitno uneti, odabrati od niza ponuđenih alternativa ili odabrati iz dve grupe ponuđenih alternativa, uparivanjem.

Alternative su opisane tekstom, a svaki mogući odgovor sadrži i informaciju o daljem toku učenja u slučaju da je učenik odgovorio na odgovarajući način ili odabrao određenu alternativu kao svoj odgovor na postavljeni zadatak.

U sistemu OSOF iskorišćena je metoda okvira za pretstavljanje ovako organizovanog znanja. Okvir se sastoji od skupa imenovanih slotova koji sadrže vrednosti ili pokazivače na druge okvire [5]. Struktura podataka koja odgovara navedenoj reprezentaciji i organizaciji znanja zapisana u pseudo jeziku za opis okvira je:

```
Begin Sekvenca
      ( Ime: Niska;
       F Početak: Članak )
Begin Članak
      ( Pojam:    Niska;
        Opis:     Niska, Simulacija, Slika;
       F Pitanje: Zadatak  )
End;
Begin Zadatak
      ( Opis:     Niska, Simulacija, Slika;
       F Sledeći: Zadatak;
       F Odgovor: Alternativa )
End;
Begin Alternativa
      ( Opis:     ( Izbor, Otvoreni odgovor,
                    Uparivanje );
       F Akcija:  ( Članak, Zadatak,
                    Dopuna_opisa, Kraj );
       F Sledeći: Alternativa )
End;
Begin Kraj
      ( Ime: Niska;
       F Akcija: ( Sekvenca, Članak, 'Exit' ) )
End;
```

U okvirima Sekvenca, Članak, Zadatak i Kraj "Niska" označava tekst proizvoljne dužine, "Simulacija" proceduru za simuliranje nekog procesa, a "Slika" proceduru za grafički prikaz proizvoljne slike. Unutar okvira Alternativa mogući zadaci koje učenik treba da izvrši su: "Izbor" - na postavljeni zadatak se odgovara izbor jedne od navedenih alternativa, "Otvoreni odgovor" - zahtev za eksplicitni upis rešenja i "Uparivanje odgovora" - postavljeni zadatak se rešava uparivanjem odgovarajućih alternnativa iz dva skupa. Na osnovu učenikovog rešenja zadatka, akcija računara se odvija prelaskom na: jedan od okvira Članak, Zadatak i Kraj, ili grupu okvira Članak koji pružaju dodatno objašnjenje nejasnog opisa ("Dopuna opisa"). Oznaka F ispred imena slota označava da je njegova vrednost pokazivač na okvir.

6. SISTEM OSOF

Sistem OSOF [3] je razvijen u Institutu za matematiku u Novom Sadu i sastoji se iz modula za prikupljanje znanja TEA, njegovu internu reprezentaciju po opisanoj šemi i dva modula elementarnog korišćenja za usvajanje znanja: izborom alternativa - LEA i testiranjem - EXA. Realizovan je programom vođenim menijima i izuzetno je jednostavan za korišćenje, te od nastavnika i učenika ne zahteva nikakvo znanje o ustrojstvu računara niti znanje programiranja. Primenom sistema zaključeno je da je odabrana metoda okvira pogodna za primenu u obrazovanju.

LITERATURA:

1. Barr A., Feigenbaum E. A., Handbook of Artificial Intelligence, Stanford University Computer Science Dept, Stanford, (1980), USA

2. Buchanan B. G., Feigenbaum E. A., Dendral and Meta-Dendral: their applications dimension, Artificial Intelligence, 11(1,2), 5-24, (1978), USA.

3. Paunić Đ., Jerinić Lj., Budimac Z., Ivanović M., Univerzalni programski paket za primenu računara u nastavi, u štampi.

4. Clancey W. J., Methodology for building an intelligent tutoring system, from "Methods and tactics in Cognitive Science", Lawrence Erlbaum Publishers, (1983), USA.

5. Frost R., Introduction to Knowledge Based Systems, Collins, London, (1986), Great Britain.

6. Ivanović M., Jerinić Lj., Paunić Đ., Budimac Z., On knowledge representation in education, Review of research, Institute of Mathematics, Novi Sad, (1988), (u štampi).

7. Nilsson N. J., Principles of Artificial Intelligence, Tioga Publishing, Palo Alto, (1980), USA.

8. Winston P., Artificial Intelligence, Addison - Wesley, Reading, (19770, USA.