Likovna umjetnost, matematika i algoritmi

Vlatko Čerić

Sadržaj

- Kratak pregled povijesti veze umjetnosti i matematike
- Matematika i računalna tehnologija u likovnoj umjetnosti
- Algoritamska umjetnost
- Neki pristupi i tehnike u algoritamskoj umjetnosti

Čovjek slika svojim mozgom, a ne svojim rukama

- Michelangelo

Stvaranje umjetničkih djela nije samo intuitivna aktivnost povezana uglavnom s emocijama

Tijekom povjesti umjetnosti razvijene su i različite racionalne metode i tehnike za stvaranje umjetničkih djela, npr.:

linearna perspektiva (Srednji vijek)
optički pribori (od 15. stoljeća, npr. *camera obscura*)

Kratak pregled povijesti veze umjetnosti i matematike

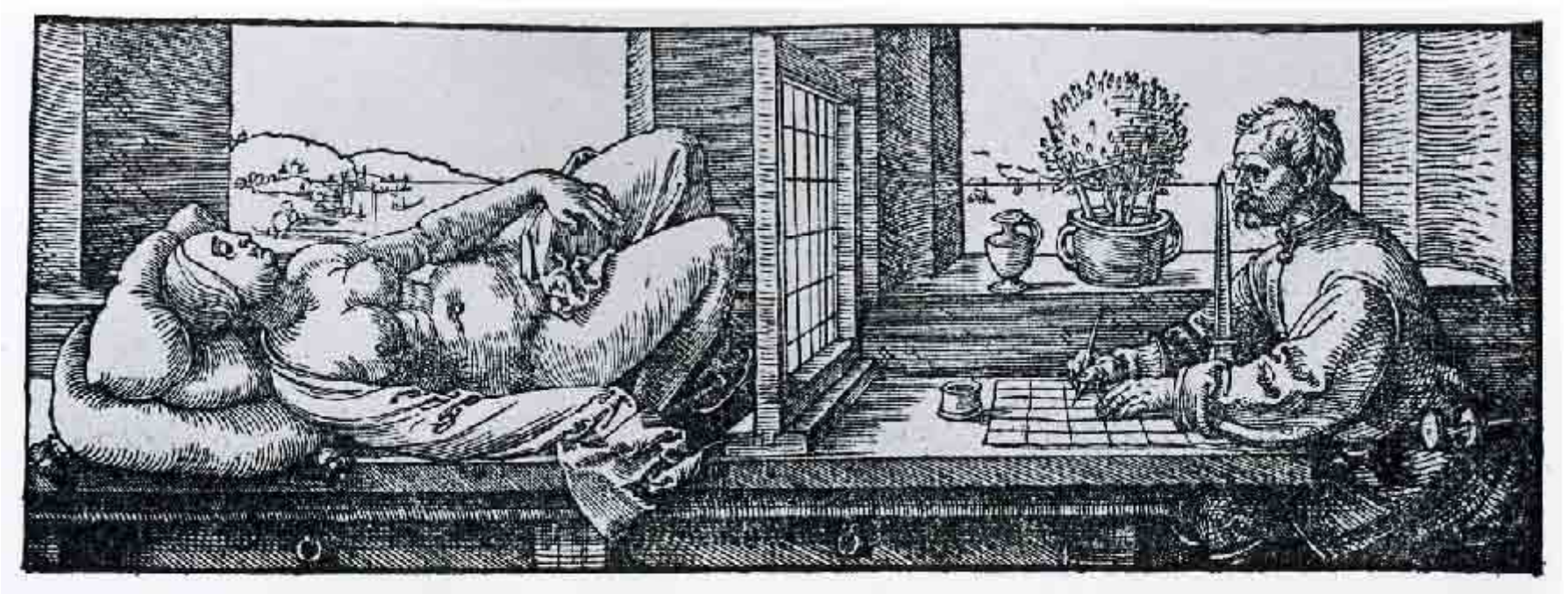
Isfahan, Iran, 15. stoljeće

podjela ravnine s nekoliko simetričnih likova

No tek su u Renesansi umjetnici počeli znatno intenzivnije koristiti znanost i matematiku.

• Leonardo da Vinci i Albrecht Dürer (15./16. stoljeće) studirali su i koristili znanje o perspektivi, proporcijama ljudskog tijela, optici i znanosti o bojama

• Piero dela Francesca (poznati slikar iz 15. stoljeća) bio je jedan od najvećih autoriteta za perspektivu. On "... je imao strast za geometrijom i planirao je sve svoje radove matematički do posljednjeg detalja."
(objavio je i nekoliko traktata iz matematike i geometrije)



Albrecht Dürer, *Artist Drawing a Nude with Perspective Device*, 1525

Najznačajniji utjecaj matematike i znanosti na umjetnost počinje u novije vrijeme, u 20. stoljeću

taj utjecaj se ne javlja samo u vizualnim umjetnostima: tako je npr. mađarski kompozitor Bela Bartok u nekim kompozicijama primjenjivao zlatni rez i Fibonaccijev niz

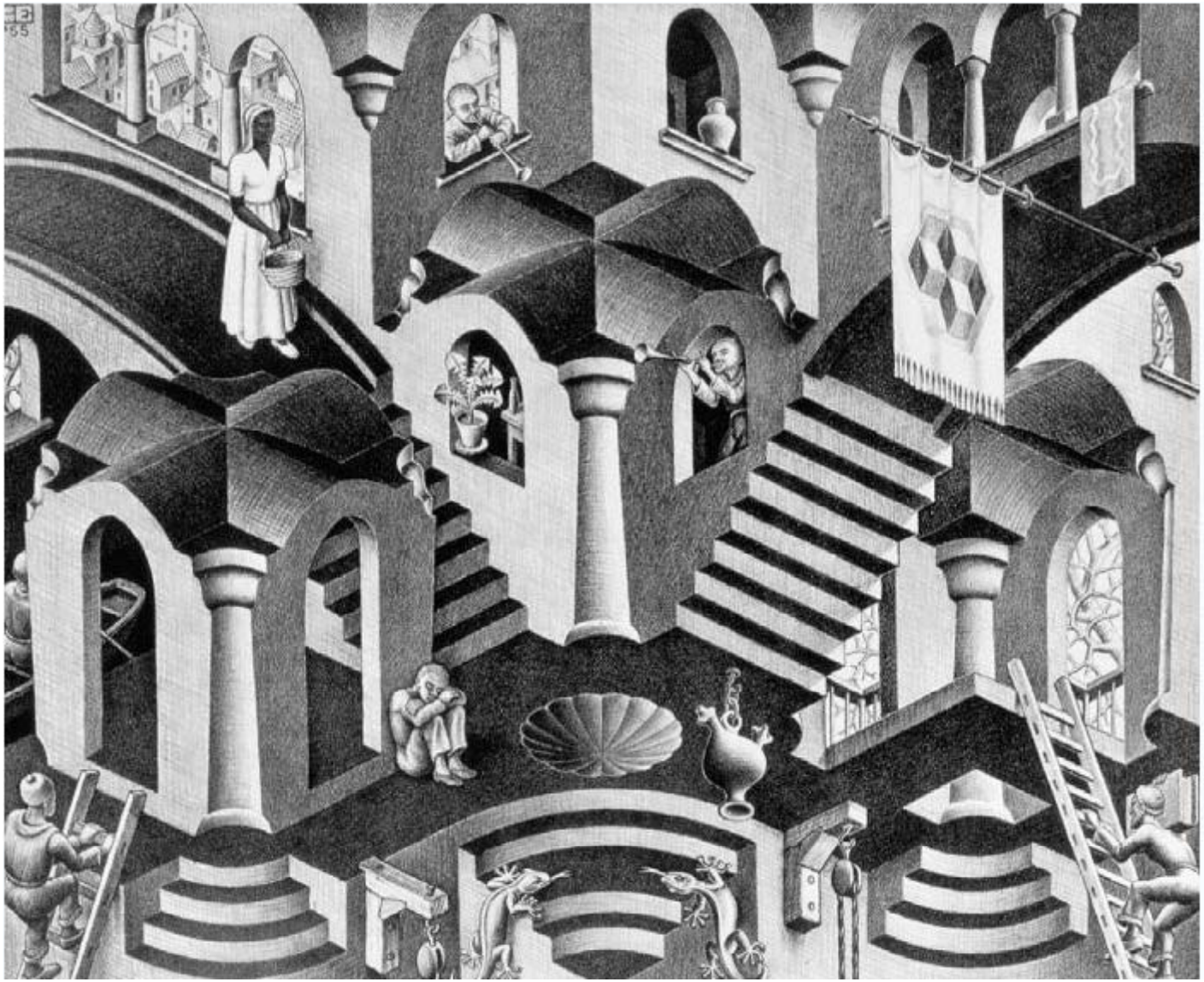
Najpoznatiji umjetnik 20. stoljeća na kojeg je utjecala matematika bio je holandski grafičar **M. C. Escher** (1898-1972).

On nije imao matematičko obrazovanje ali je bio zainteresiran za matematiku i na njega je utjecao matematičar Roger Penrose.

Escher je studirao:

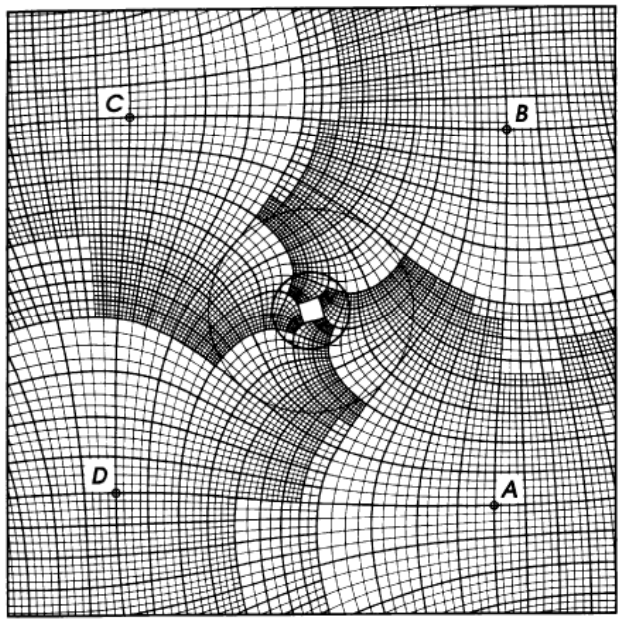
- neobične perspektive i projekcije na sferi
- korištenje teselacija ravnine (pravilne podjele ravnine)
- različite transformacije oblika, itd.

Prvu fazu stvaranja svojih radova Escher je posvećivao razvoju geometrijskog modela slike



M.C. Escher, *Convex and Concave*, 1955

M.C. Escher, *Print Gallery*, 1956



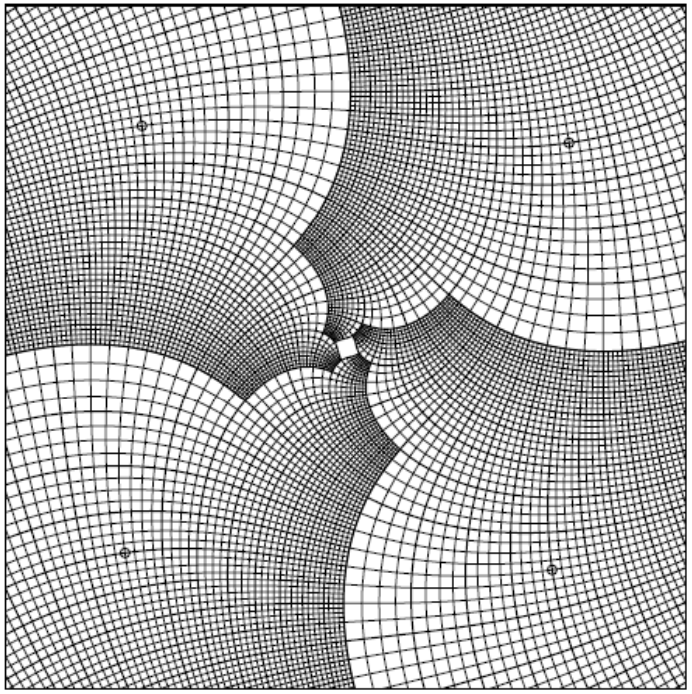
Escherov
geometrijski
model slike

Matematička rekonstrukcija grafike: “ispunjavanje praznine u središtu slike”

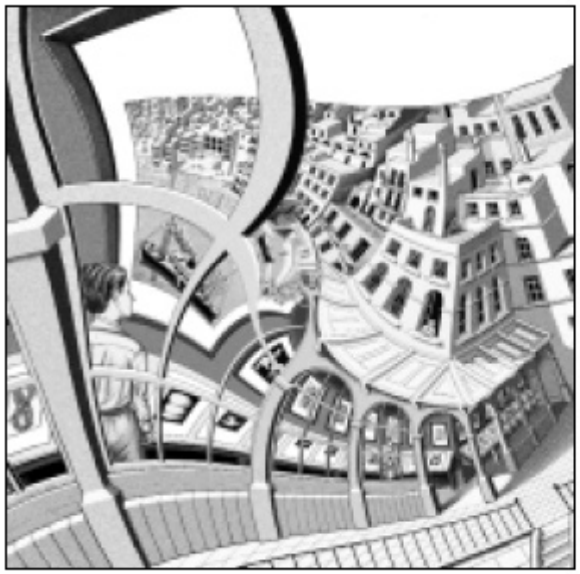
(B. de Smit and H. W. Lenstra Jr., 2003)

Grafika se može gledati kao da je nacrtana na određenoj eliptičkoj krivulji nad poljem kompleksnih brojeva - idealizirana verzija slike ponavlja se u središtu

Idealna konformna mreža – Escher je bio vrlo blizu nje



Slike dobivene računalnim programom koji sadrži matematičku rekonstrukcije ove grafike



središte slike

središte povećano 4x

središte povećano 16x

Matematika i računalna tehnologija u likovnoj umjetnosti

Razvoj računalne tehnologije i posebno grafičkih kartica i softvera u drugoj polovini 20. stoljeća napravio je ogroman utjecaj na mogućnost vizualizacije u matematici

i korištenje vizualizacije u dizajnu i umjetnosti

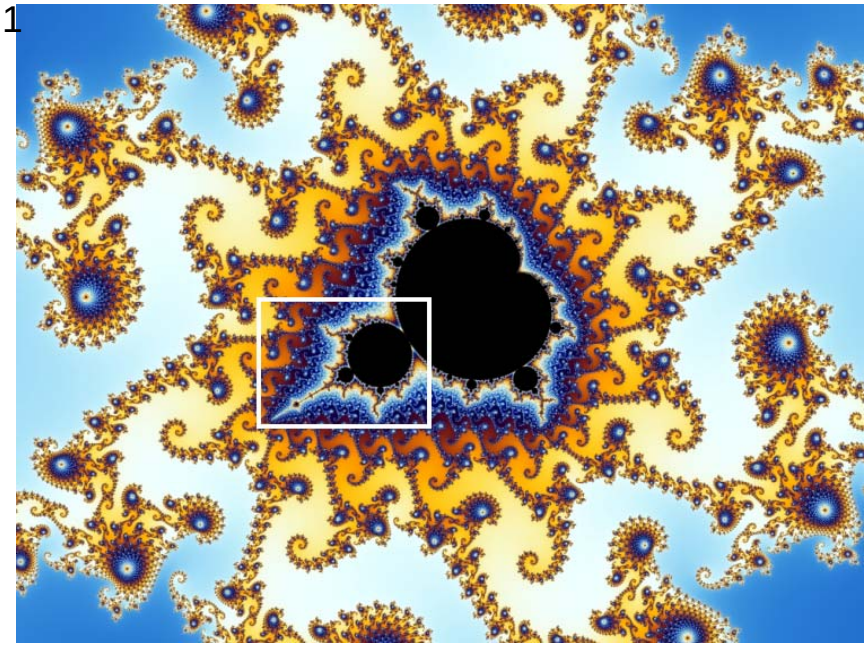
Jedan od najpoznatijih pristupa pogodnih za stvaranje atraktivnih vizualnih struktura su **fraktali** (Benoit Mandelbrot),

porodica objekata koji su sebi slični i invarijantni na veličinu

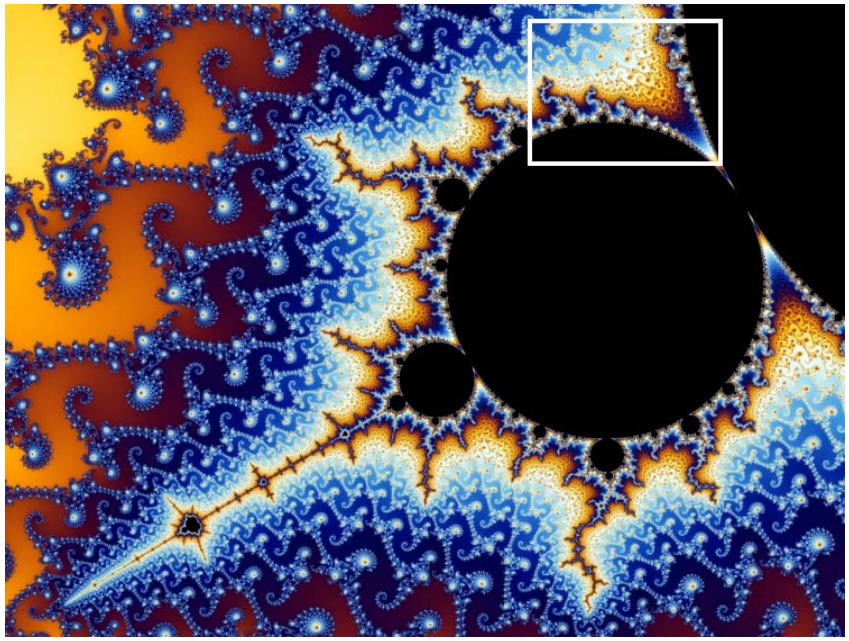
te imaju jednostavnu rekurzivnu definiciju

Zanimljivi primjeri fraktala su Mandelbrotov skup i Julia skup. Mandelbrotov skup definira se pomoću porodice kompleksnih kvadratičnih polinoma.

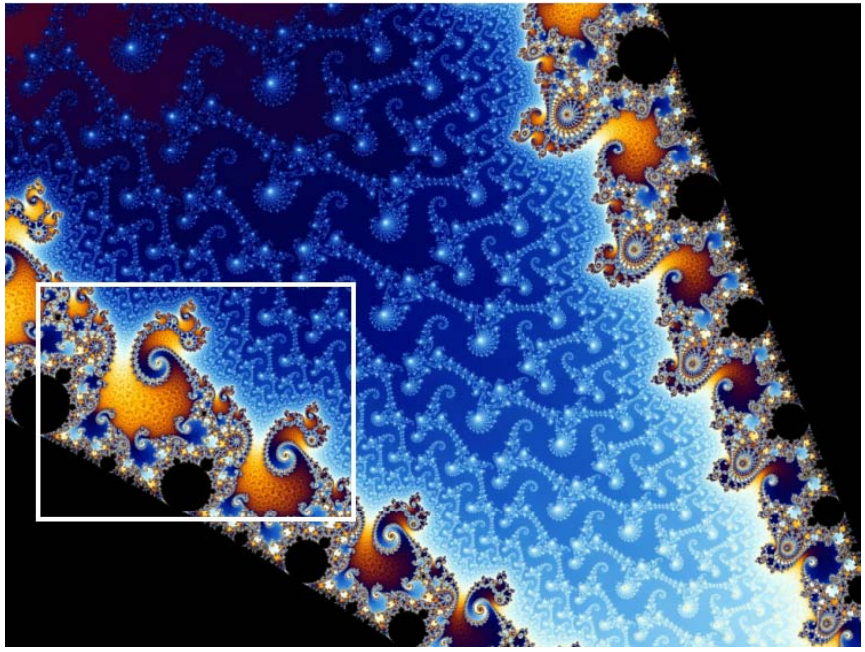
1



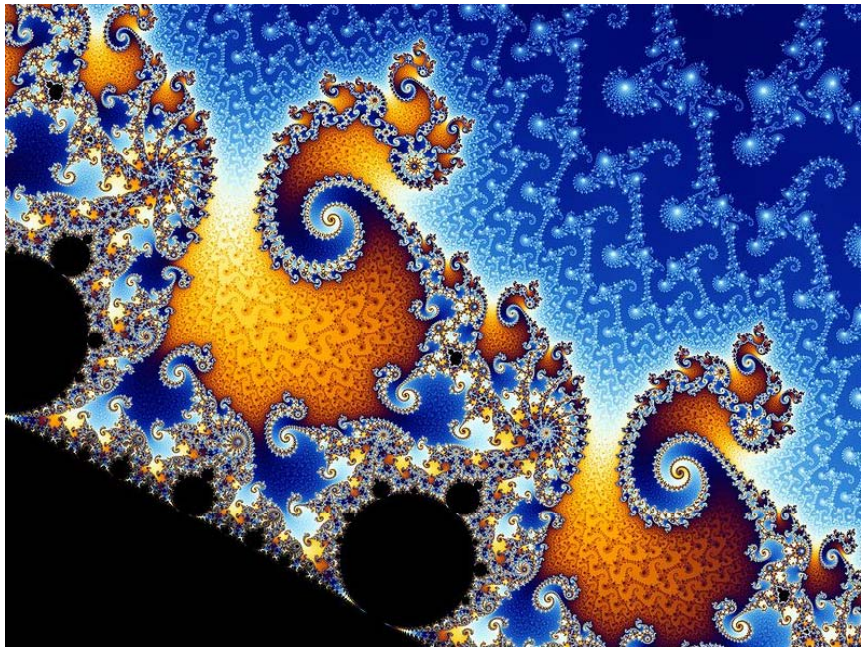
2



3



4



Za generiranje atraktivnih slika koriste se i druge matematički utemeljene metode, npr. **genetski algoritmi** i **stanični automati** (*cellular automata*).

Genetski algoritmi

- nasumično generiranje prve generacije slika
- izbor najuspjelijih slika
- stvaranje novih generacija slika međusobnim križanjem u mutacijom najboljih primjeraka prethodne generacije slika

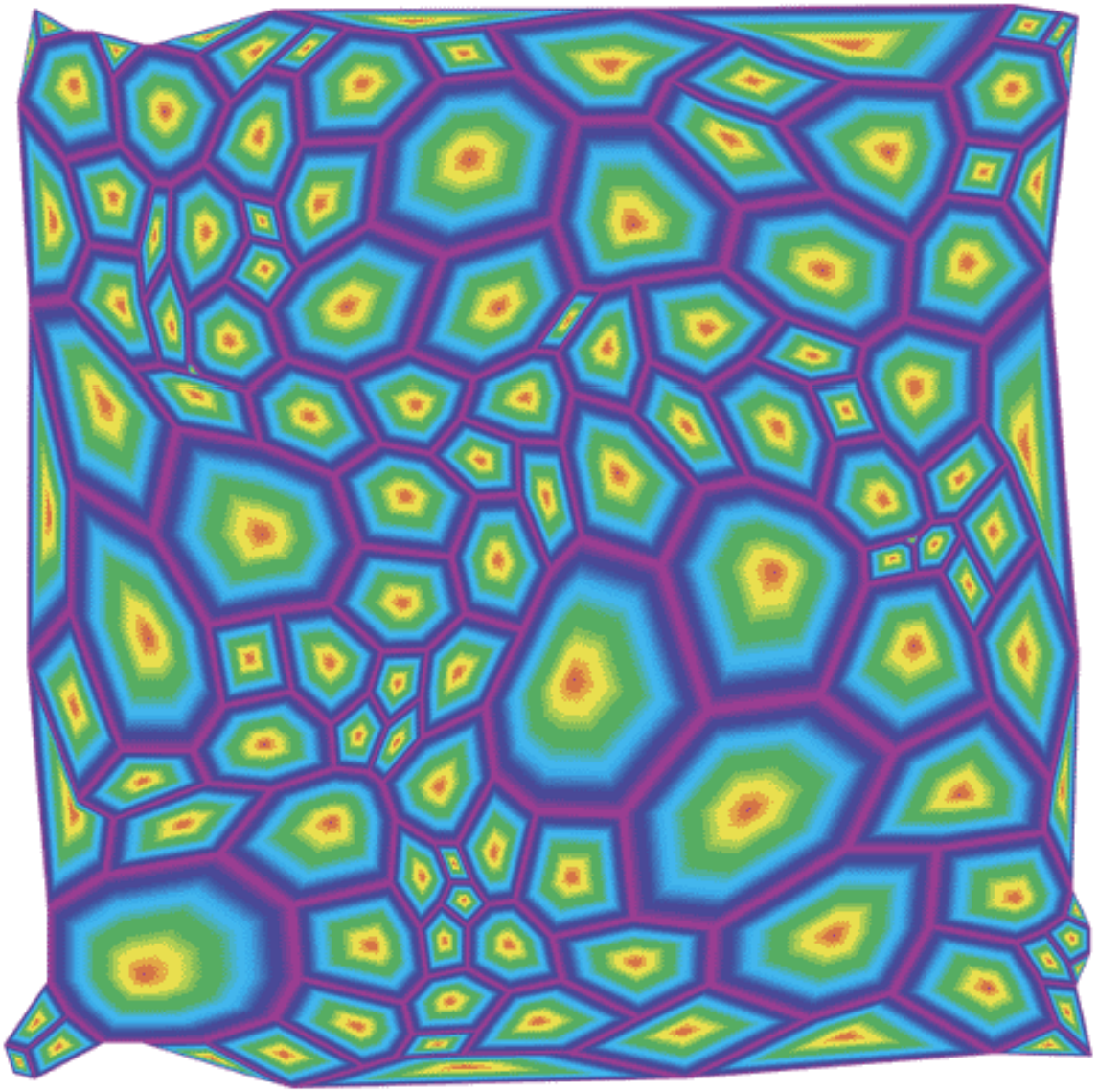
Stanični automati

- jednostavni apstraktni sustavi pravila
- mreže stanica: stanice se razvijaju u vremenu u ovisnosti o stanjima svojih najbližih susjeda

➢ Vizualizacija matematičkih objekata

➢ Michael Trott: vizualizacija matematičkih objekata korištenjem softvera *Mathematica*.

Random Colored Voronoi Tesselation



Created by: Michael Trott

- Korištenje matematike kao alata za razvoj umjetnikovih ideja

Najčešće se koristi dosta jednostavna matematika koja omogućuje realizaciju ideja umjetnika

Potrebno je puno eksperimentiranja kako bi se postigli vizualno interesantni rezultati.

Ovaj pristup koristi se u **algoritamskoj umjetnosti**.

Algoritamska umjetnost

osnovni pojmovi

algoritam je precizna procedura za rješavanje nekog problema

algoritam kodiran u nekom programskom jeziku je računalni program

algoritamska umjetnost – izvođenje računalnih programa koji sadrže algoritme za generiranje slika

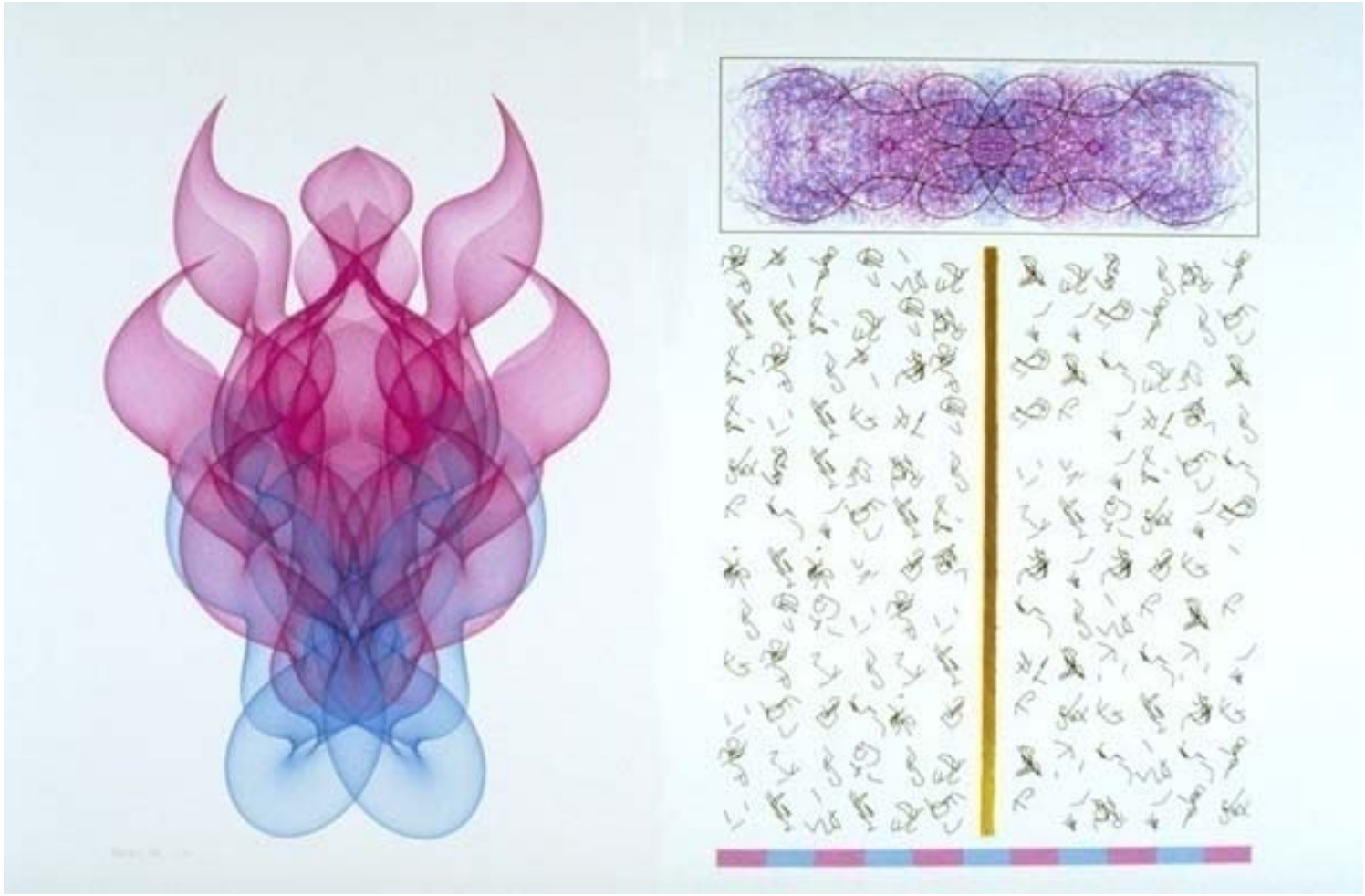
algoritamska umjetnost – slike se generiraju na temelju algoritama koji u potpunosti opisuju način njihova nastanka

ti algoritmi opisuju strukturu i boje slike,
te način kako će ona biti generirana
(npr. korake u stvaranju slike na ekranu ili na štampaču)

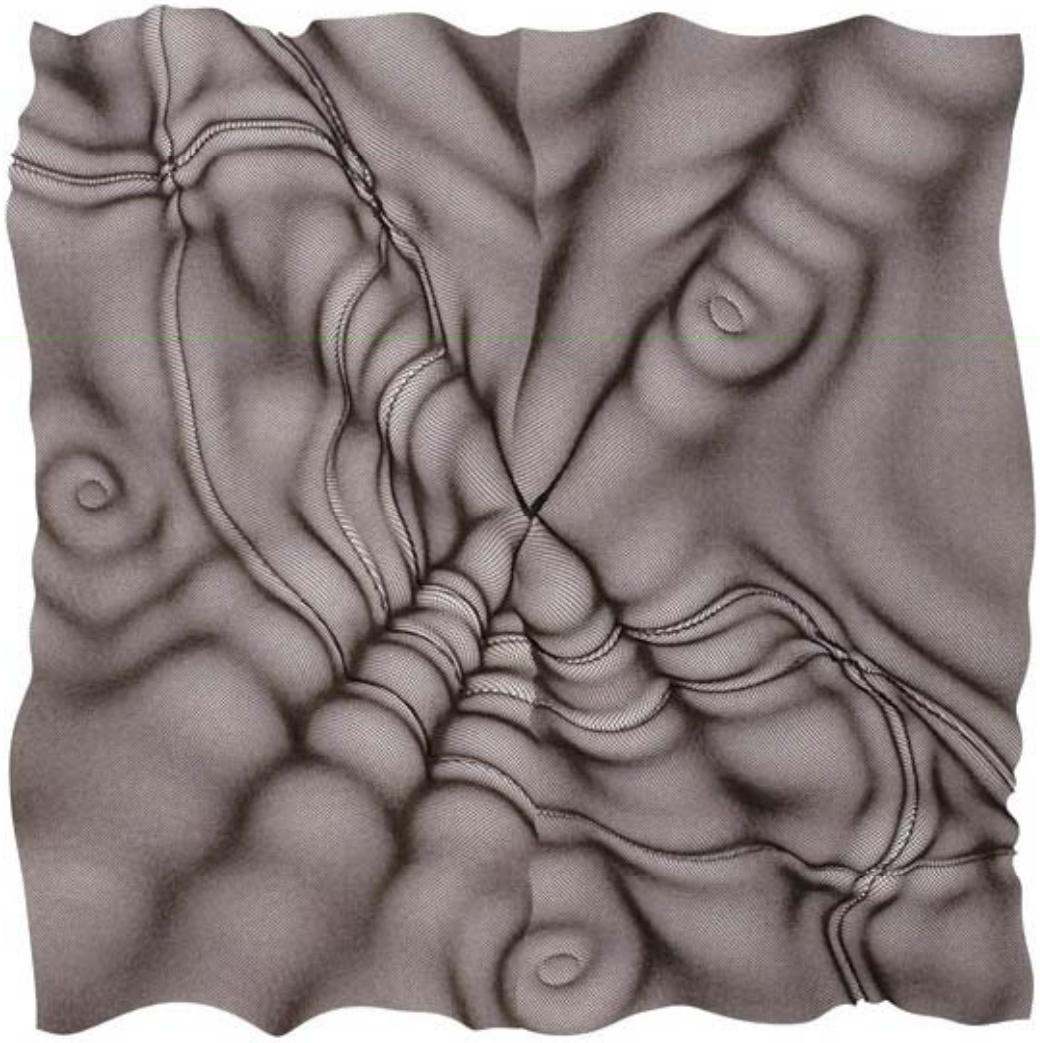
programi za generiranje slike sadrže:

- autorovu **ideju** o tome kako bi slika trebala izgledati,
- i **tehniku** pomoću koje se ona može pretvoriti u sliku

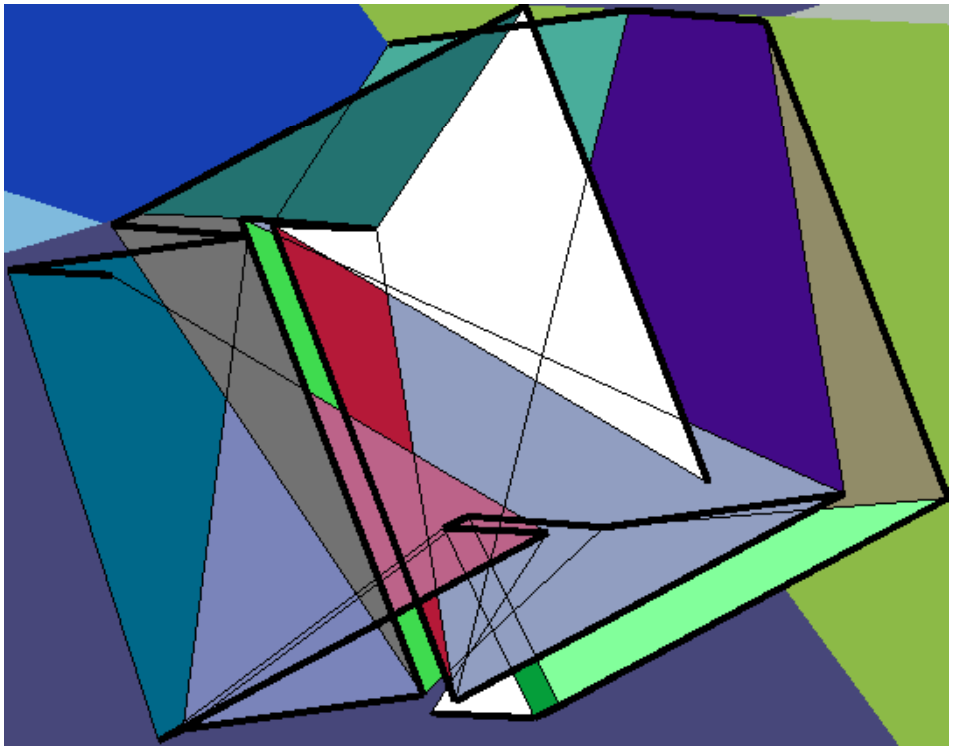
Nekoliko poznatih algoritamskih umjetnika



Roman Verostko, *Burning bush*, 2000



Jean-Pierre Hébert, *vent noire II*, 1989



Manfred Mohr, *P-702/F*, 2000 (projekcija 6-dimenzionalne hiperkocke)

Korištenje algoritamskog pristupa i odgovarajućih računalnih progama omogućuje

- stvaranje kompleksnih slika
- veliku preciznost i brzinu realizacije
- eksperimentiranje s nizom alternativnih struktura i koloristićkih rješenja

Budući se algoritamska slika stvara pomoću računala,
da li njen autor uopće treba koristiti bilo kakve estetske procjene?

- Ništa se ne može stvoriti bez početne ideje o djelu

- a autor neprestano procjenjuje slike dobivene eksperimentiranjem s programom (algoritmom)

  i na temelju procjene mijenja algoritam (program)
  sve dok ne dobije zadovoljavajući rezultat

To ujedno govori o specifičnom karakteru algoritamske umjetnosti:

- autor mora posjedovati racionalne sposobnosti potrebne da razvije algoritam i napiše odgovarajući programski kôd
(semantika i sintaksa kôda),

- a ujedno mora imati intuitivne i estetske sposobnosti neophodne da smisli zanimljiv pristup i sadržaj novih radova, napravi dobar izbor vizualno zanimljivih generiranih slika (*images*) i odredi obećavajući smjer daljnjeg rada.

Neki pristupi i tehnike u algoritamskoj umjetnosti

preokupiraju me sljedeće teme

i njihov utjecaj na estetiku umjetničkih djela

- linearnost vs. nelinearnost
- determinizam vs. slučajnost (nasumičnost)

  kontrolirana slučajnost

  (različita razina slučajnosti na različitim područjima slike)
- jednostavnost vs. kompleksnost

  koherentna kompleksnost

  (matematičko-algoritamsko modeliranje omogućuje postavljanje principa kojim se stvara kompletna slika)

pristupi i tehnike koje sam koristim

razvijeno je mnogo različitih pristupa i tehnika,
ali o onom što sam radim mogu dati najviše detalja

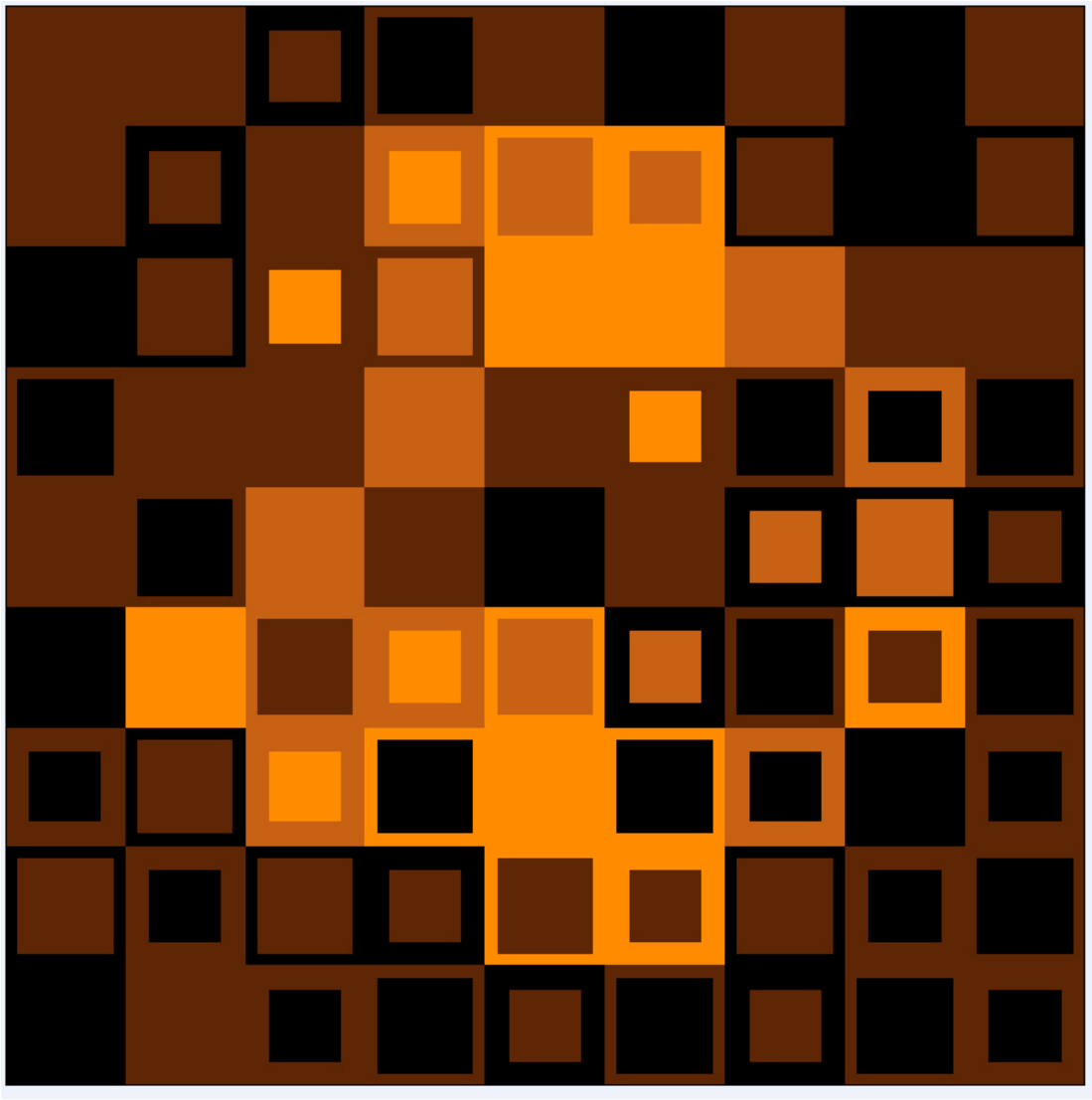
- konstruktivni pristup
- matematičko modeliranje
- algoritamska manipulacija slika
- animacija
- skulpture

konstruktivni pristup

- nešto jednostavnije slike: realizacija unaprijed zamišljene strukture

  (često su to geometrijske strukture)

- ili njihove modifikacije



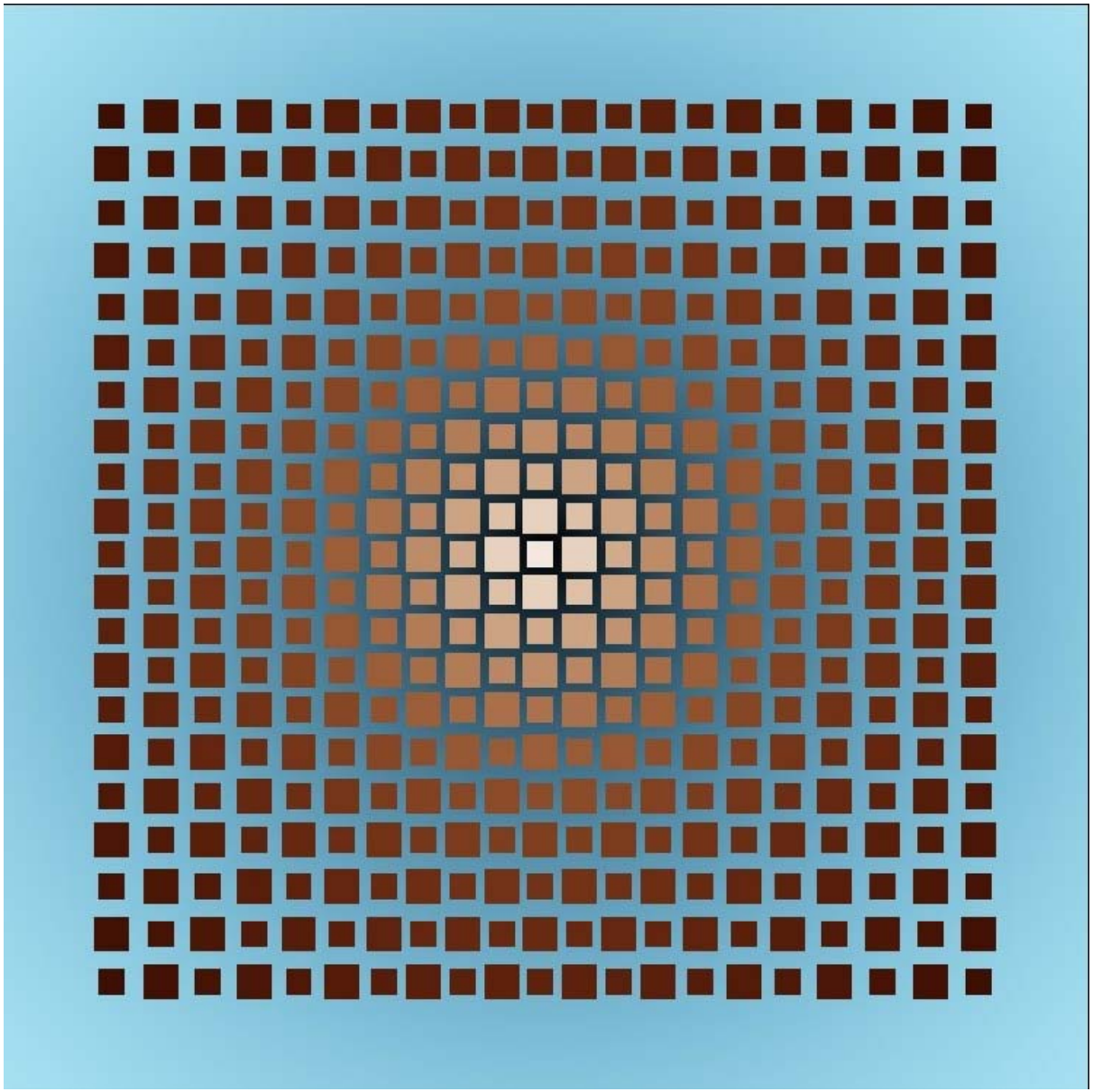
V. Čerić, Composition 8, 2009

Osnovni elementi ovog ciklusa grafika:

- koristi se ograničeni skup boja
- koriste se kvadrati triju veličina
  (veličina rastera i dvije manje veličine)
- postoje dva sloja na koje se stavljaju kvadrati:
  podloga i slika koja se stavlja na podlogu
- cijela površina slike dijeli se na centralni i vanjski dio
  (centralni dio može biti ograničen krugom, kvadratiom i sl.)

  centralni i vanjski dio površine slike koriste različit skup boja kvadrata podloge i slike na podlozi
- na podlozi se stavljaju samo kvadrati veličine rastera slike, a na slici nad podlogom može biti samo jedan od dvaju manjih kvadrata (jedna i druga veličina su naizmjenično raspoređene po plohi)
- izbor boja i veličina kvadrata (na slici nad podlogom) izvodi se nasumičnim (slučajnim) odabirom

Informatička tvrtka IN2, Zagreb



kompliciraniji slučaj konstruktivne metode

V. Čerić, Order 1, 2007

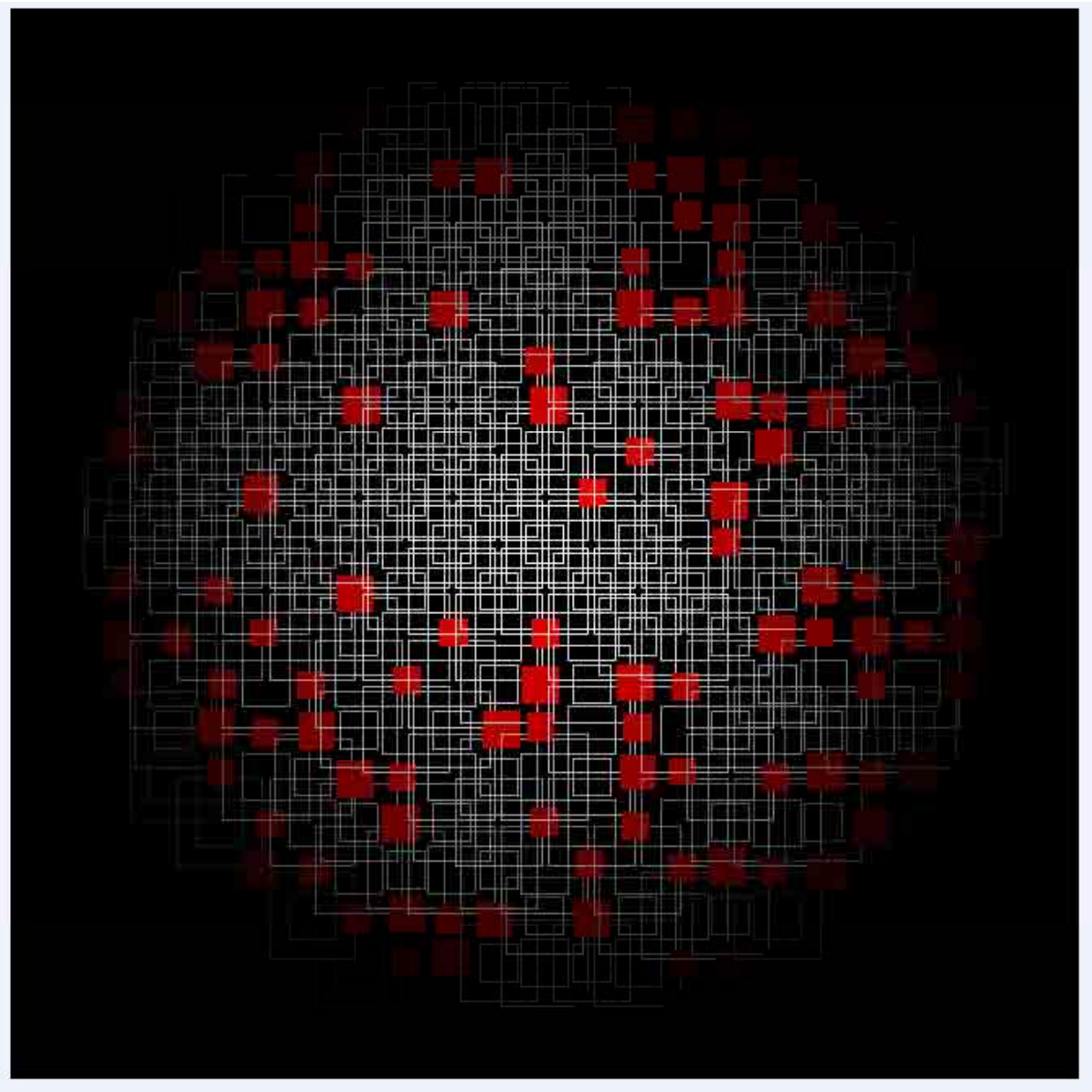
Ovo je nešto složeniji slučaj u kojem:

- postoje dva sloja na koje se stavljaju kvadrati:
  podloga i slika koja se stavlja na podlogu
- podloga čini kvadrat veličine cijele slike

  boja podloge se kontinuirano mijenja od svijetle boje na rubovima do tamne u centru slike
  (da se jače istaknu svijetli kvadrati u centru na slici iznad podloge)
- na slici nad podlogom koriste se kvadrati dviju veličina
  (jedna i druga veličina su naizmjenično raspoređene po plohi)

  boja kvadrata slike se kontinuirano mijenja od tamne boje na rubovima do svijetle u centru slike

  međusobna udaljenost centara kvadrata se smanjuje od rubova slike prema centru (kako bi centar slike što više svijetlio)
- sve je deterministički određeno, tj. nema nasumičnogm (slučajnog) odabira ni jedne veličine

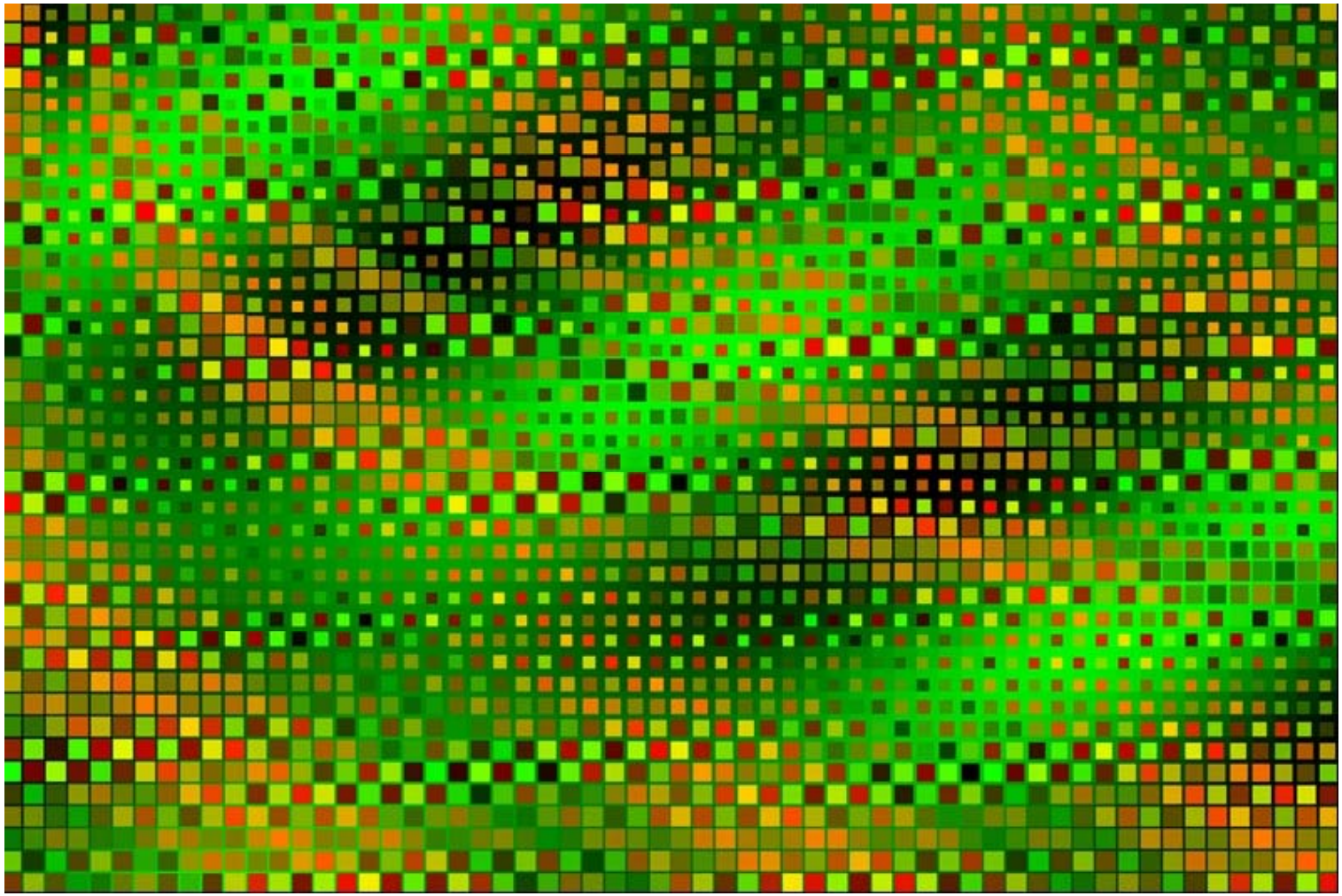


dodatni primjeri konstruktivne metode

igra kvadrata i linija

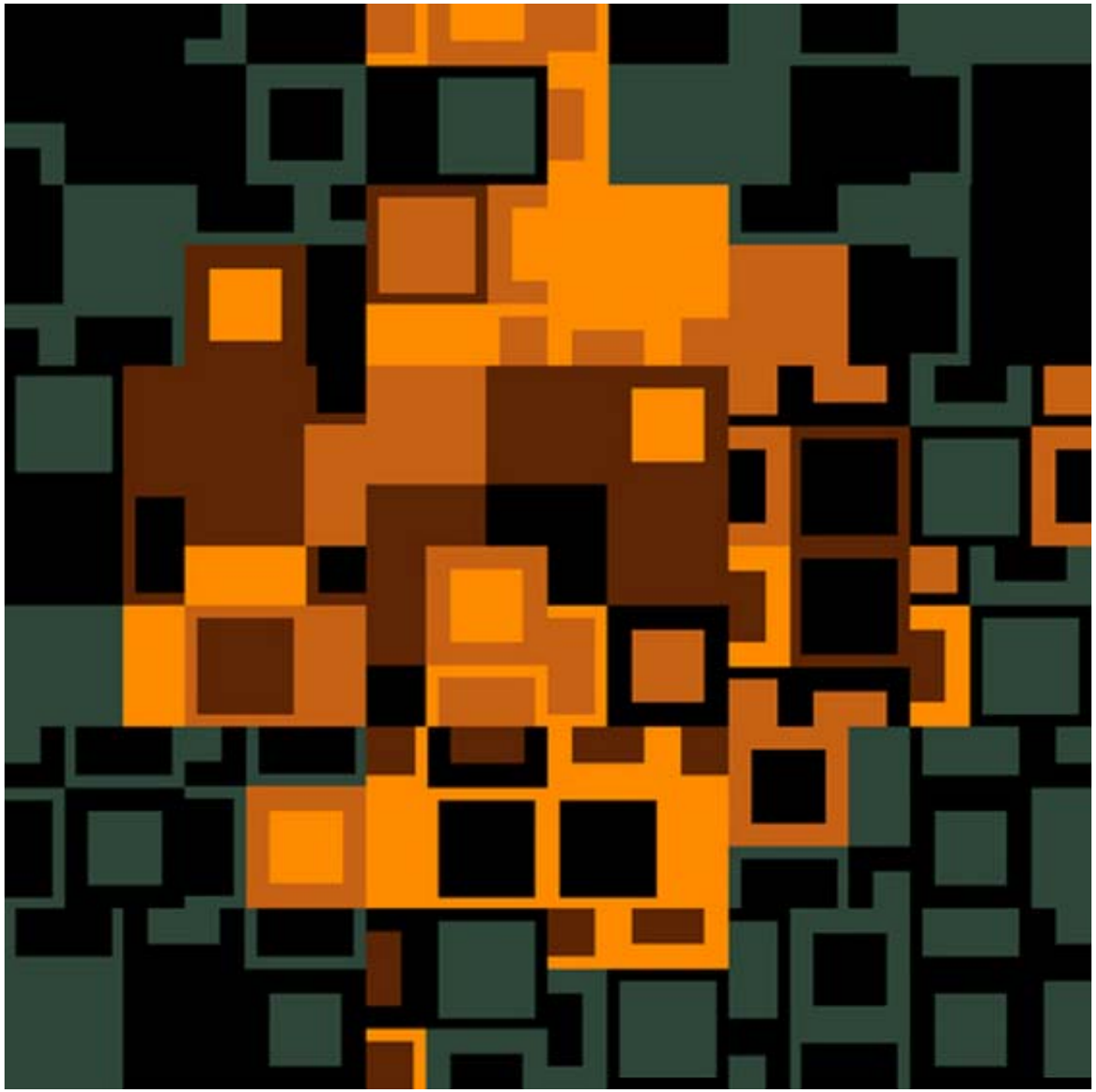
V. Čerić, Agglomeration 34, 2007

široki raspon veličina kvadrata i njihovih boja



V. Čerić, , *Ocean 14*, 2010

fragmentacija slike



V. Čerić, *Fragmentation 2*, 2012

Fragmentacija slike - jedna od staza prema kompleksnosti:

- bazira se na dekonstrukciji slike,
  i to bilo neke harmonične geometrijske forme ili neke fotografije.
- dekonstrukcija se izvodi podjelom originalne slike u mrežu kvadrata
- ti se kvadrati zatim rotiraju korištenjem bilo determinističkog bilo slučajnog pristupa

Osnovni elementi algoritma:

```
SeedRandom[8908]

...

kvadrat = npix;
   (*  broj pixela slike koja se rastavlja u "ploćice" *)
 brojploc = nploc;   (*  broj "pločica" na koji se slika dijeli *)
  ploc = IntegerPart[ kvadrat / brojploc ];
    (*  dužina brida "pločice" - u pixelima  *)

...

tmpRGB = ToRGBColor[tmp];   (*   "tmp" - učitana slika  *)
{r, g, b} = ToChannels[tmpRGB];
      (*   RGB kanali uzetog dijela fotke  *)

rr = r[[1]]; gg = g[[1]]; bb = b[[1]];
      (*  uzimanje raw image po kanalima boje  *)

Do[  sl1[[i, j]] = {rr[[i, j]], gg[[i, j]], bb[[i, j]]};
            , {i, 1, kvadrat}, {j, 1, kvadrat}  ];
     (*  "sl1" - pikseli slike sa RGB kanalima boje  *)
```

```
(* petlja po kvadratima ("plocicama") na koje se dijeli slika  i, j *)

Do[
 {

  imgpart1 = Take[ sl1,  {1 + (i - 1) * ploc, i * ploc},
    {1 + (j - 1) * ploc, j * ploc } ,  { 1, 3} ],
    (*  "imgpart1" - kvadrat (i,j) /plocica/ cijele slike  *)

  ...
  sluc = Random[Integer, {1, 4}];
      (*  konstanta za svaku "plocicu" (i,j)  *)

  Do[   imgpart[[ ii, jj ]] =  Which [
                                      sluc == 1, imgpart1[[ ii, jj ]],
                                      sluc == 2, imgpart1[[ jj, ploc - ii + 1 ]],
                                      sluc == 3,
     imgpart1[[ ploc - ii + 1, ploc - jj + 1 ]],
                                      True,
     imgpart1[[ ploc - jj + 1, ploc - ii + 1 ]]
                                    ],
               {ii, 1, ploc}, {jj, 1, ploc} ],

  ...
 }
   , {i, 1, brojploc}, {j, 1, brojploc}
 ]
```

nasumična rotacija "plocica"

petlja po dijelovima ("plocicama") slike

matematičko modeliranje

- slike definirane matematičkim modelom
- kompleksnije slike: autor ima samo približnu ideju o tome kako će slika izgledati

U ranim radovima (sredina 1970tih) koristio se linijski štampač: mogli su se koristiti samo diskretni simboli

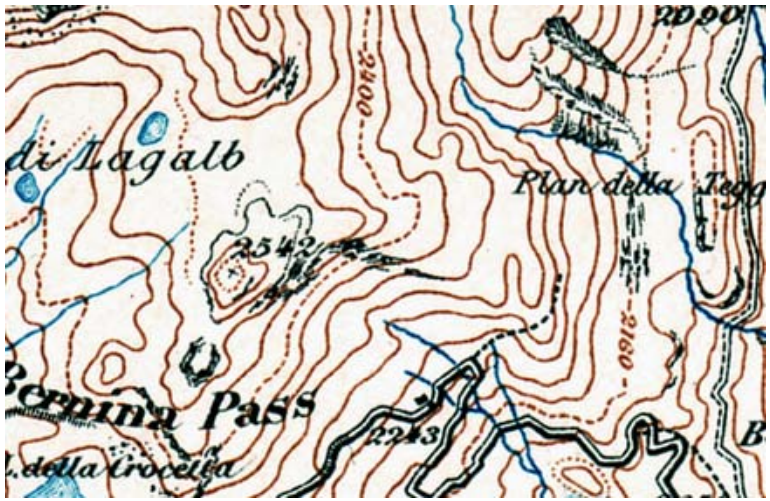
(slova, specialni znakovi, brojevi)

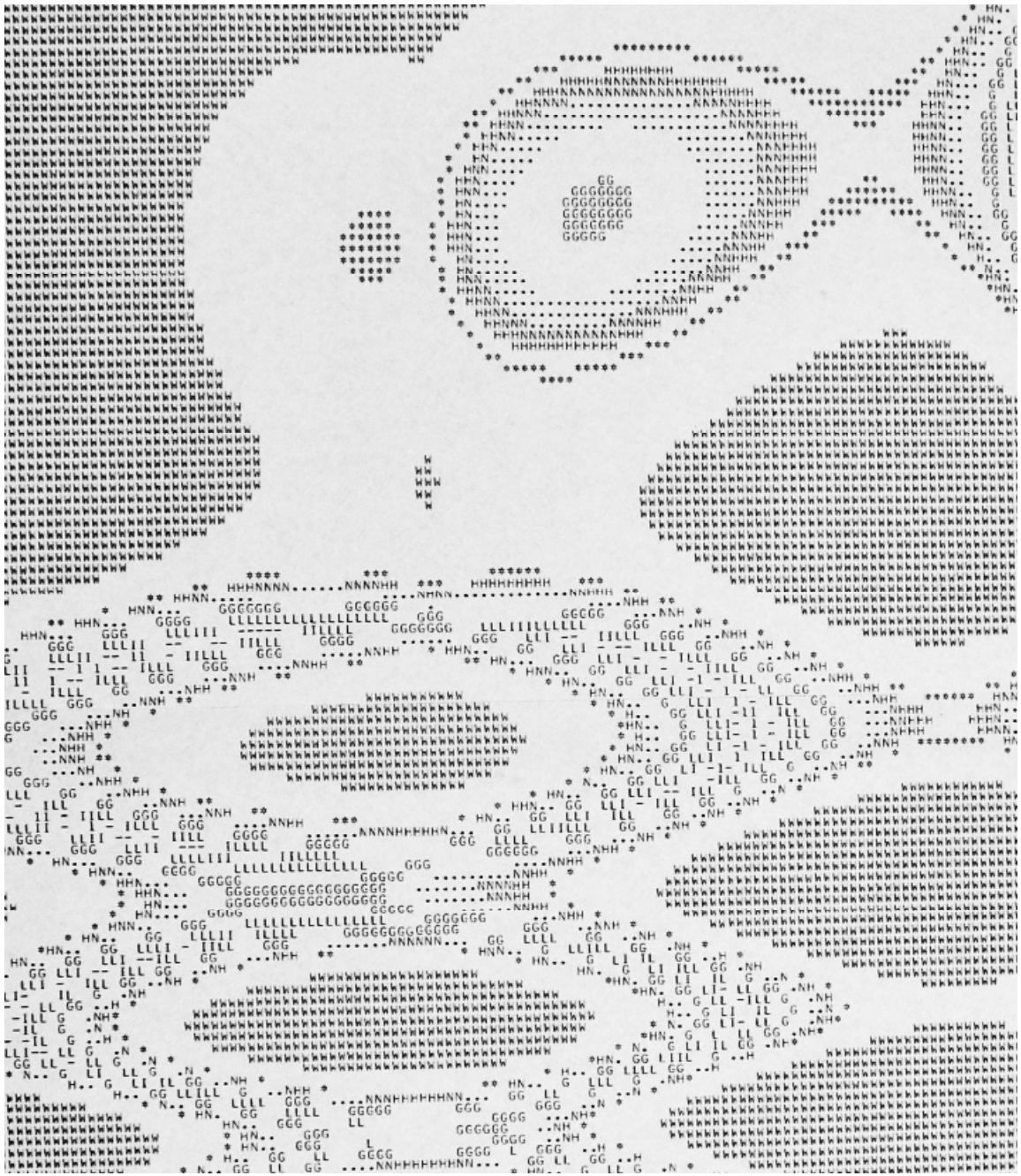
- definira se funkcija nad ravninom f(x,y)

  (npr. visina terena, izohipse tlaka i sl.)

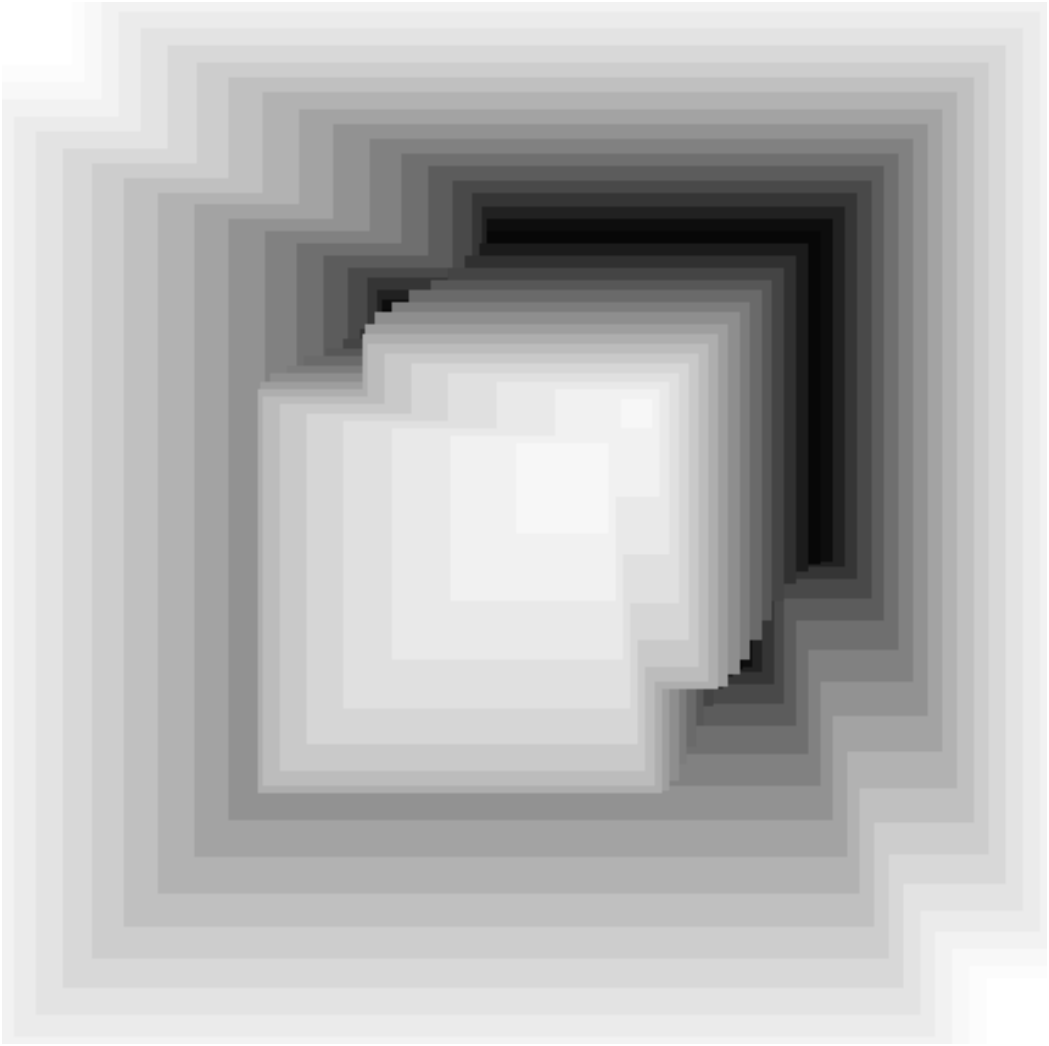
- napravi se izbor simbola koji se koriste za svaku površinu između uzastopnih izohipsi

(programski jezik Fortran, računalo IBM 1130)





V. Čerić,
Bez naslova,
1975



V. Čerić, Unclassified objects 2, 2006

Ciklus *Unclassified objects* predstavlja geometrijske strukture u monokromatskim sivim tonovima.

Ciklus je baziran na ideji uzastopnog prekrivanja ravnine s pravokutnicima

pri čemu su položaji pravokutnika, njihovi oblici, veličine i nivoi sive boje

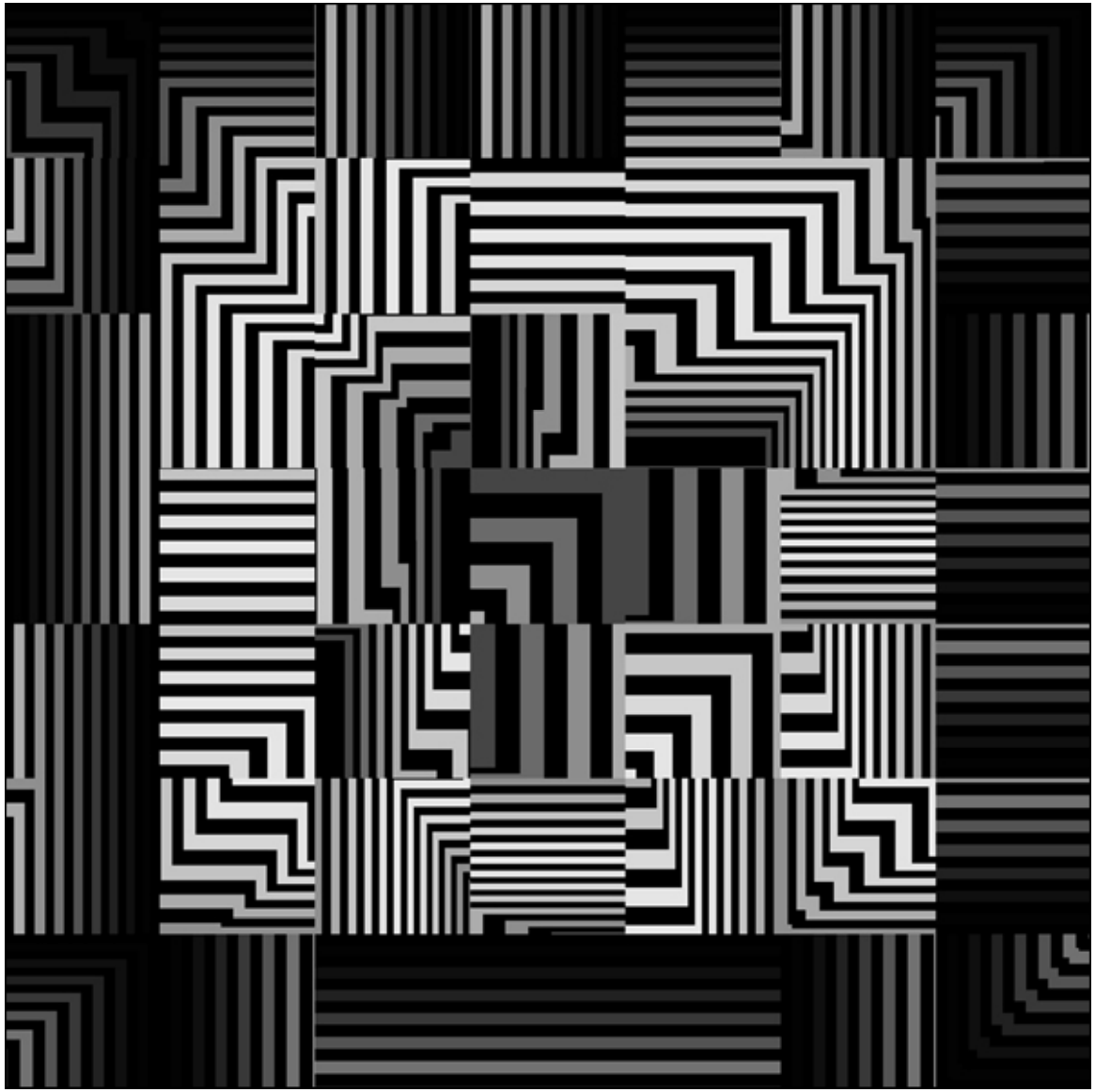
definirani matematičkim modelima temeljenim na parametarskim jednadžbama koje koriste periodičke funkcije

Primjer parametarske jednadžbe:

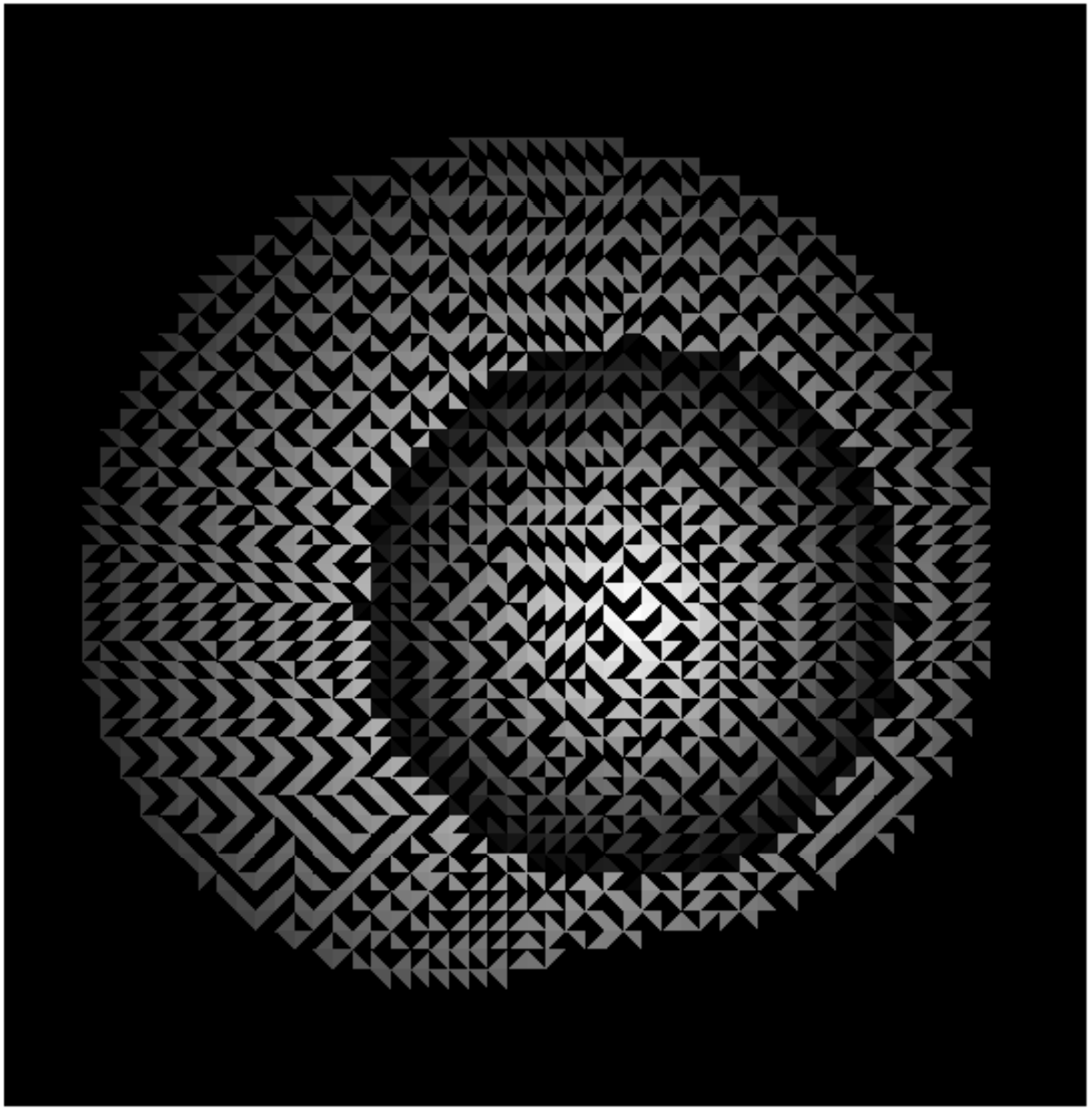
$$x = a \cos t$$
$$y = b \sin t.$$

jednadžba elipse

parametarske jednadžbe + fragmentacija slike

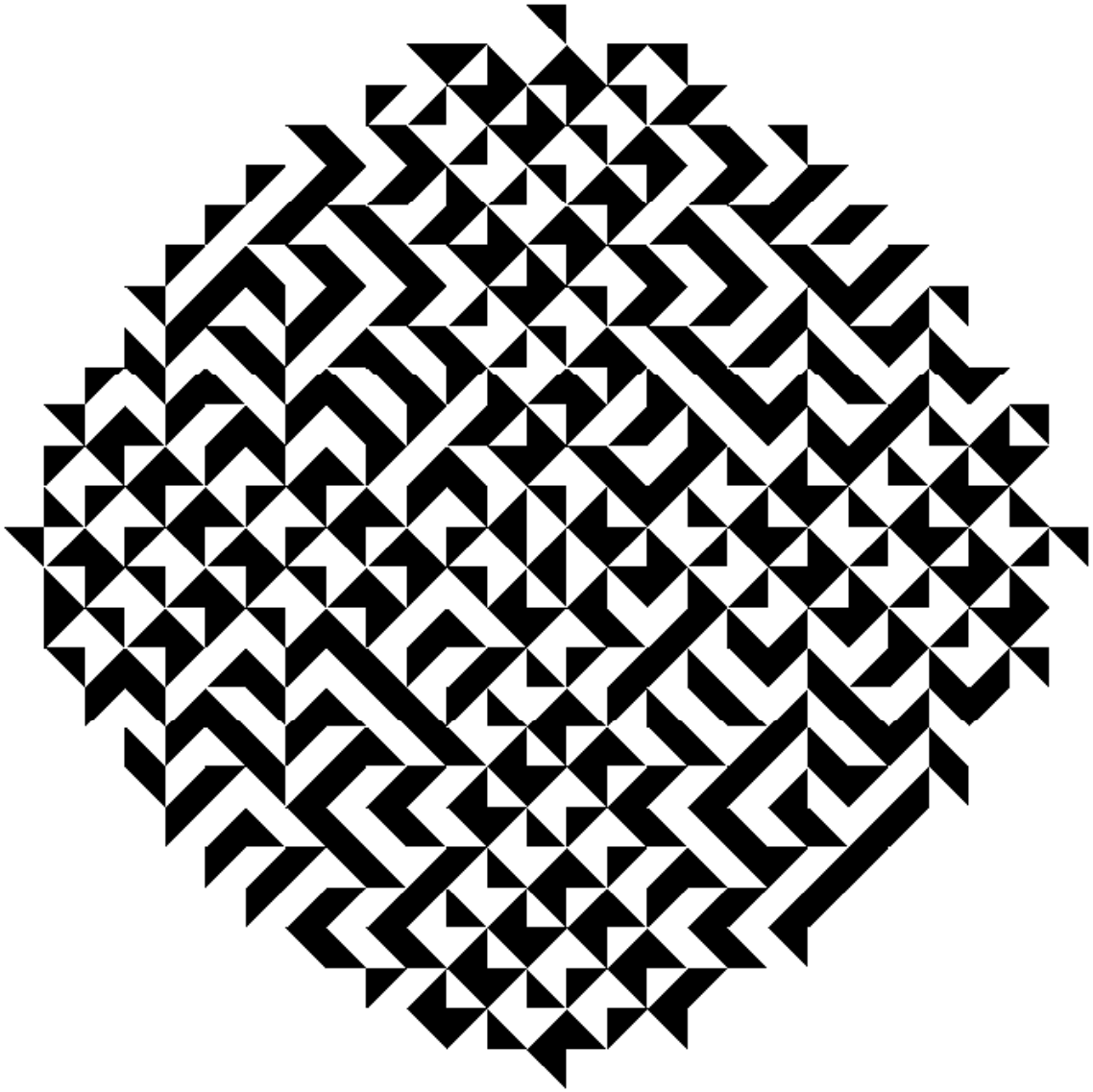


V. Čerić, *Labyrinth 7*, 2011



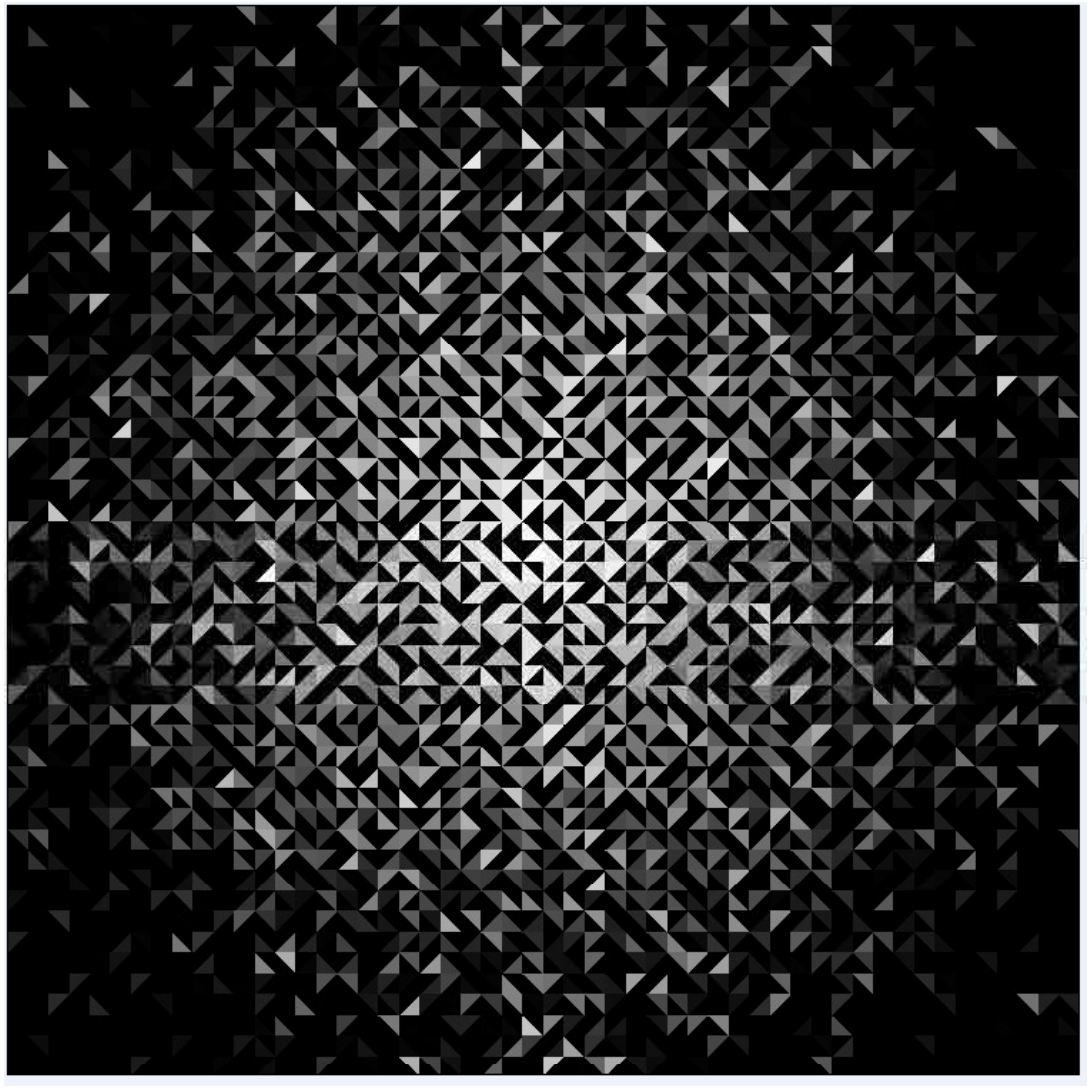
V. Čerić,
Nexus 13,
2007

Op art varijacija



V. Čerić, *Bez naslova*, 2007

Kontrolirana slučajnost (promjene razine slučajnosti na slici)

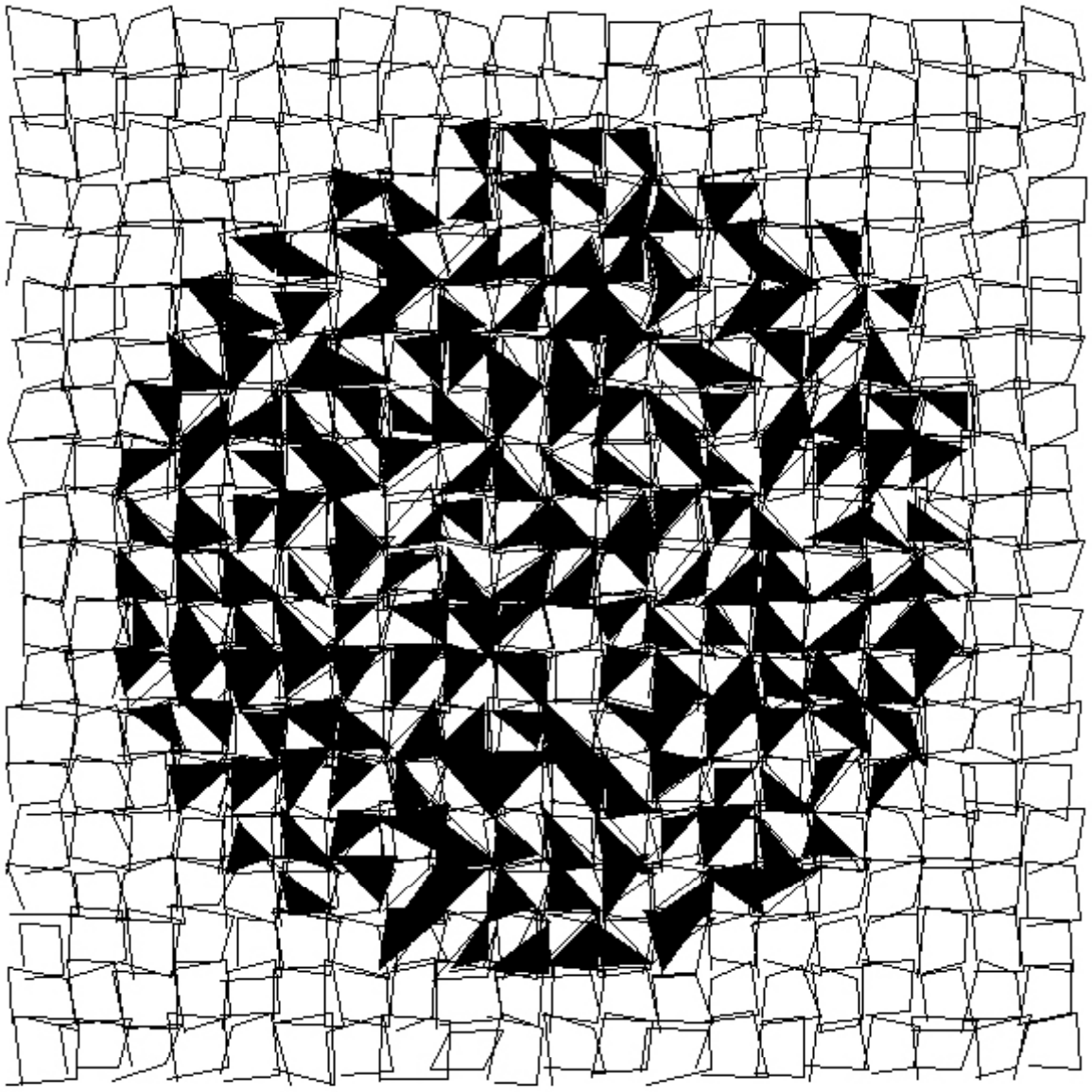


Razina slučajnosti raste od centra prema periferiji slike

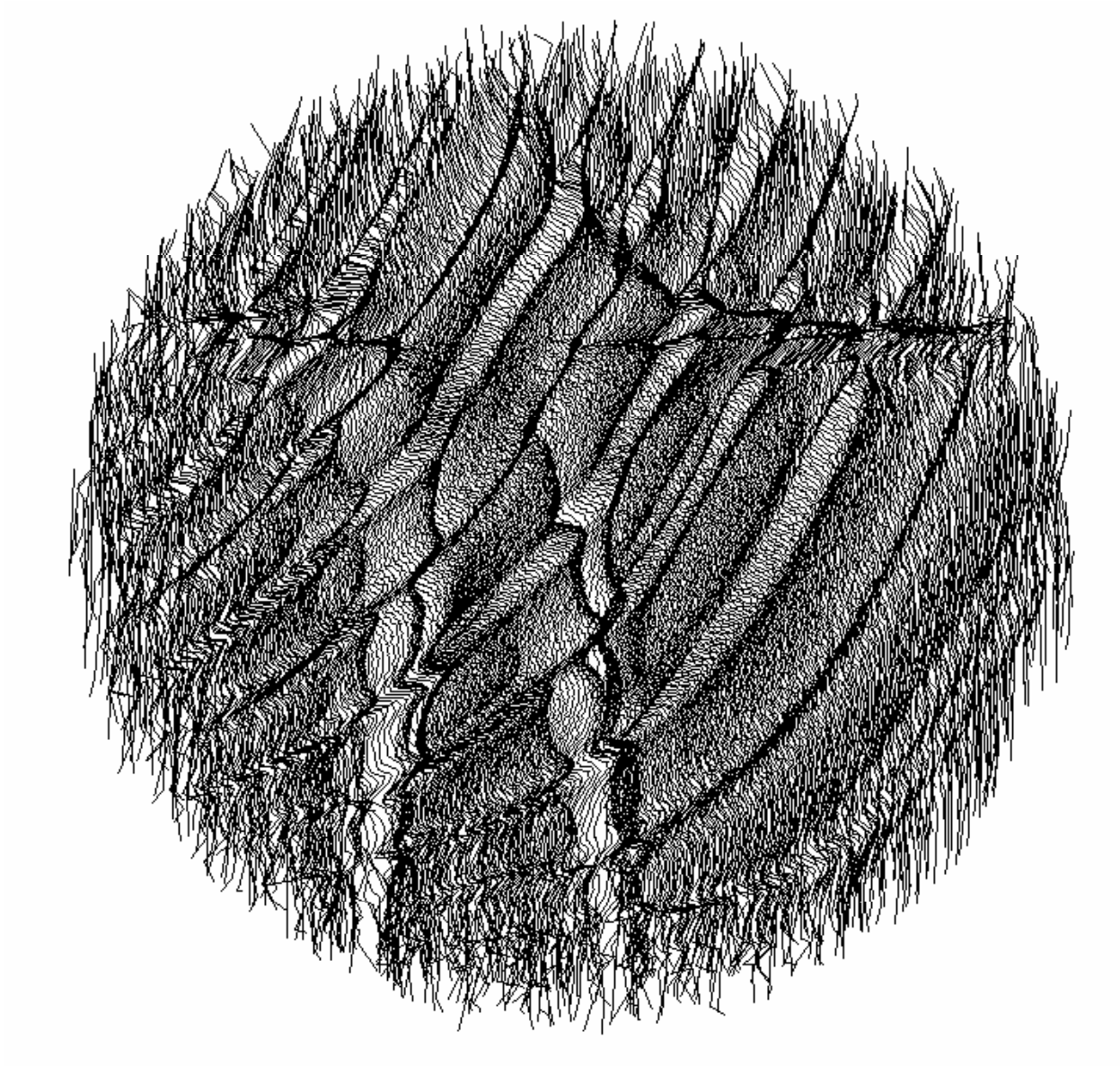
V. Čerić, Nexus 19, 2007

algoritamski "crteži"

- slobodno crtanje geometrijskih kompozicija



V. Čerić, *Bez naslova*, 2009



V. Čerić, *Evolucija 5*, 2005

algoritamska manipulacija slika (fotografija)

transformacija figurativnih oblika u apstraktne kompozicije

- Fotografije sadrže izuzetno bogatstvo spektra boja i strukture svijeta koji nas okružuje

  posljedica: algoritamske slike koje možemo dobiti na taj način imaju potencija da budu vrlo kompleksne, zanimljive a često i intrigantne

- Novo stvorene slike sadrže identične piksele kao i fotografija iz koje su nastale, i to transformirane ili regrupirane na neočekivni način koristeći algoritme i računala

Tako i iz figurativne fotografije možemo dobiti apstraktne slike

primjer linearne transformacije:

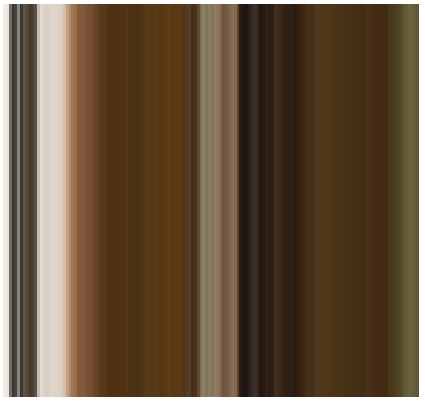
- fotografija se transformira u matricu piksela
- iz nje se transformacijom dobiva nova slika

tako da se za svaku vrijednost indeksa *j* nove slike izabere isti piksel(i,j0) (tj. ista boja)

pa se boje mijenjaju samo s promjenom koord. *i*

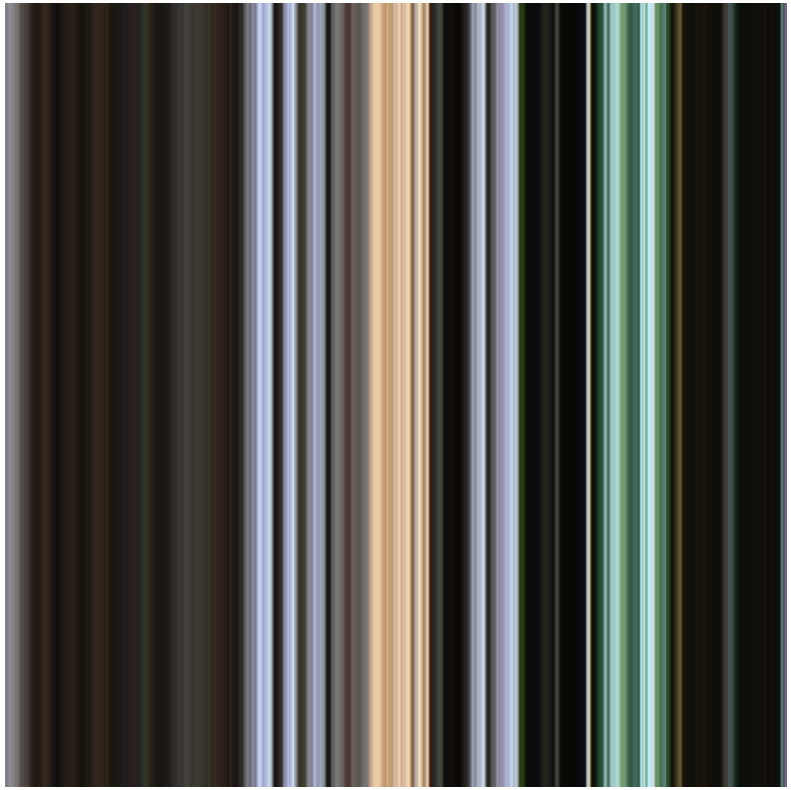
```
Do[  new ⟦ i, j ⟧  = original ⟦ i, j0 ⟧ ,
          {i, 1, n}, {j, 1, n}
   ],
```

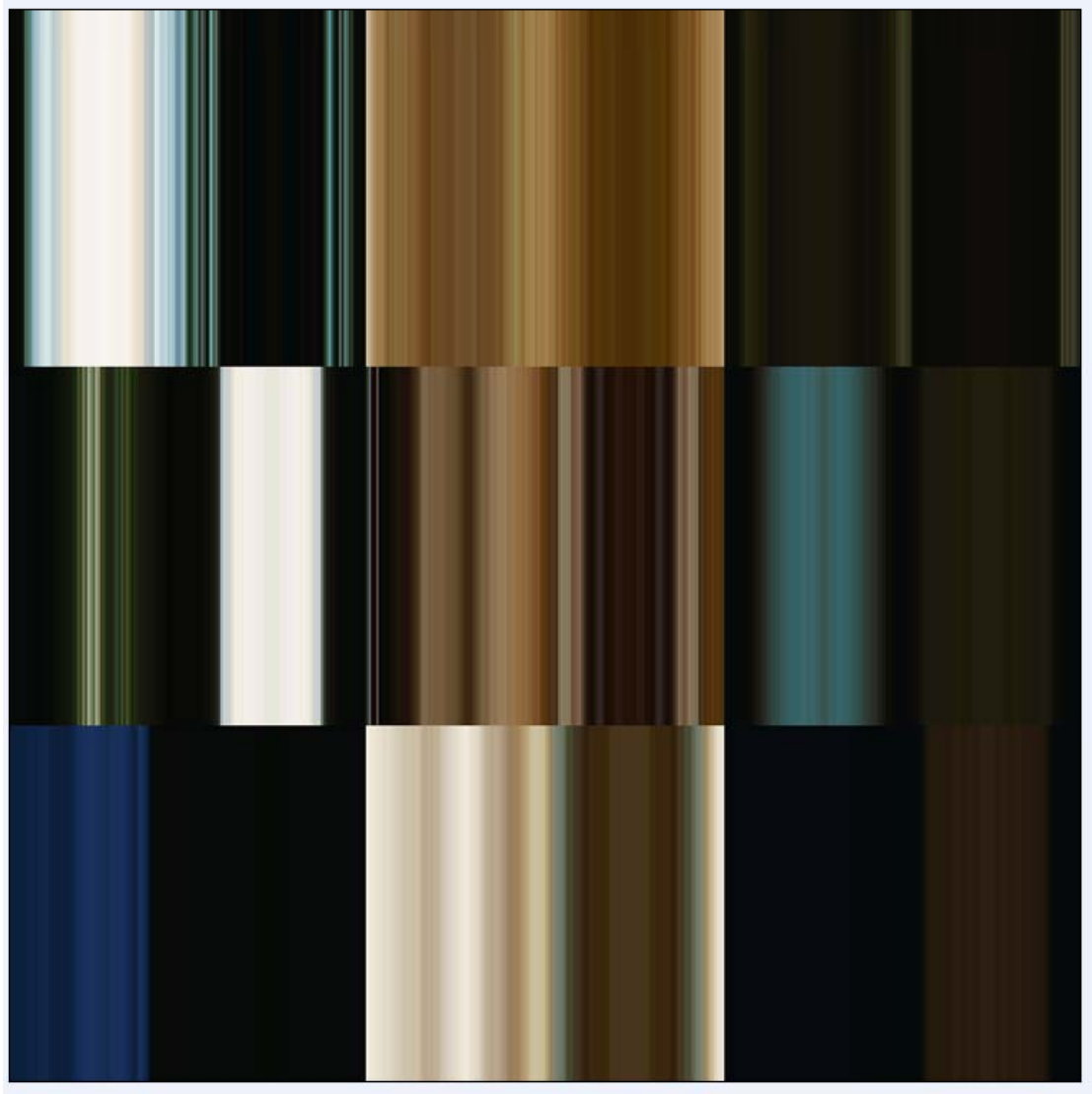
pikseli fotografije (**original**) i slike stvorene transformacijom fotografije (**new**)



Uz veličinu slika 600 x 600 piksela

```
Do[      new 〚 i, j 〛 = original 〚 i, 250〛 ,
          {i, 1, 600}, {j, 1, 600}
   ],
```





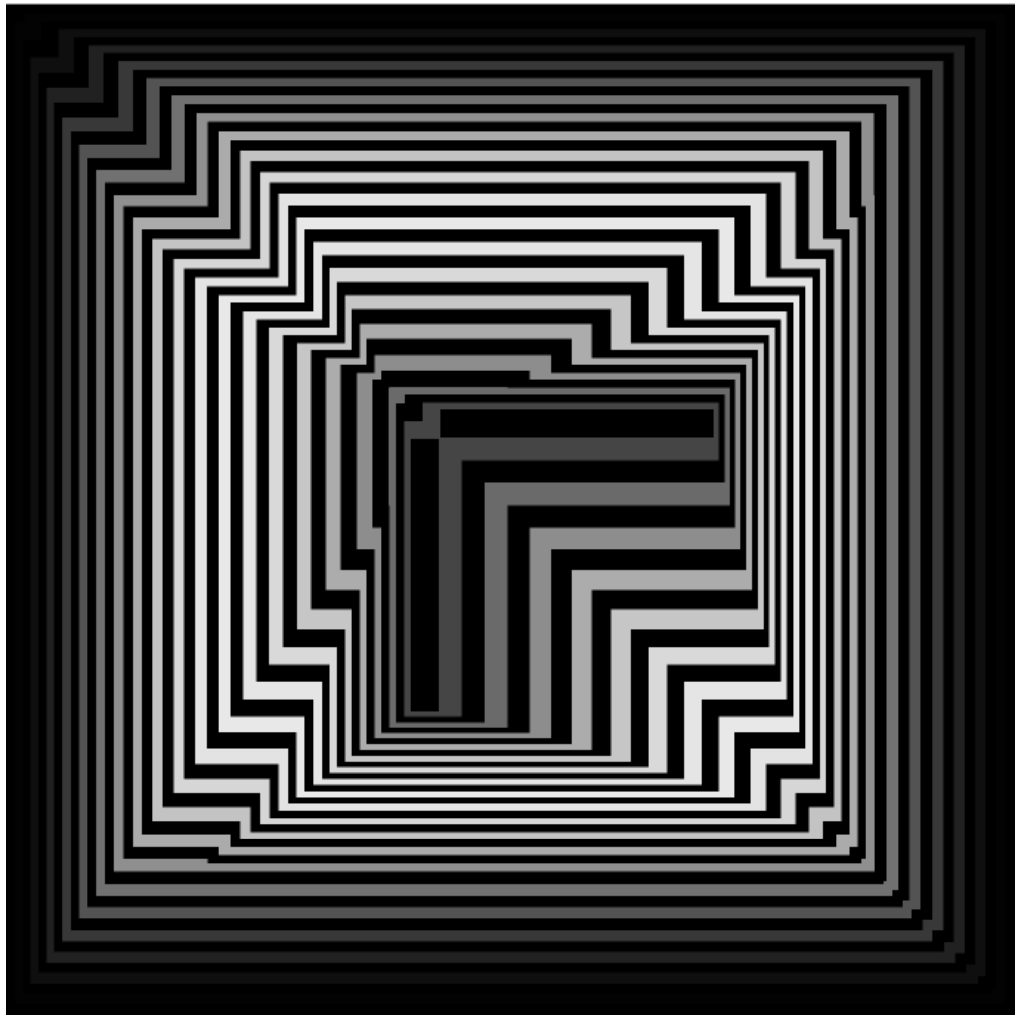
V. Čerić, *Spectral variations 10*, 2007

sekvencijalno generiranje slika

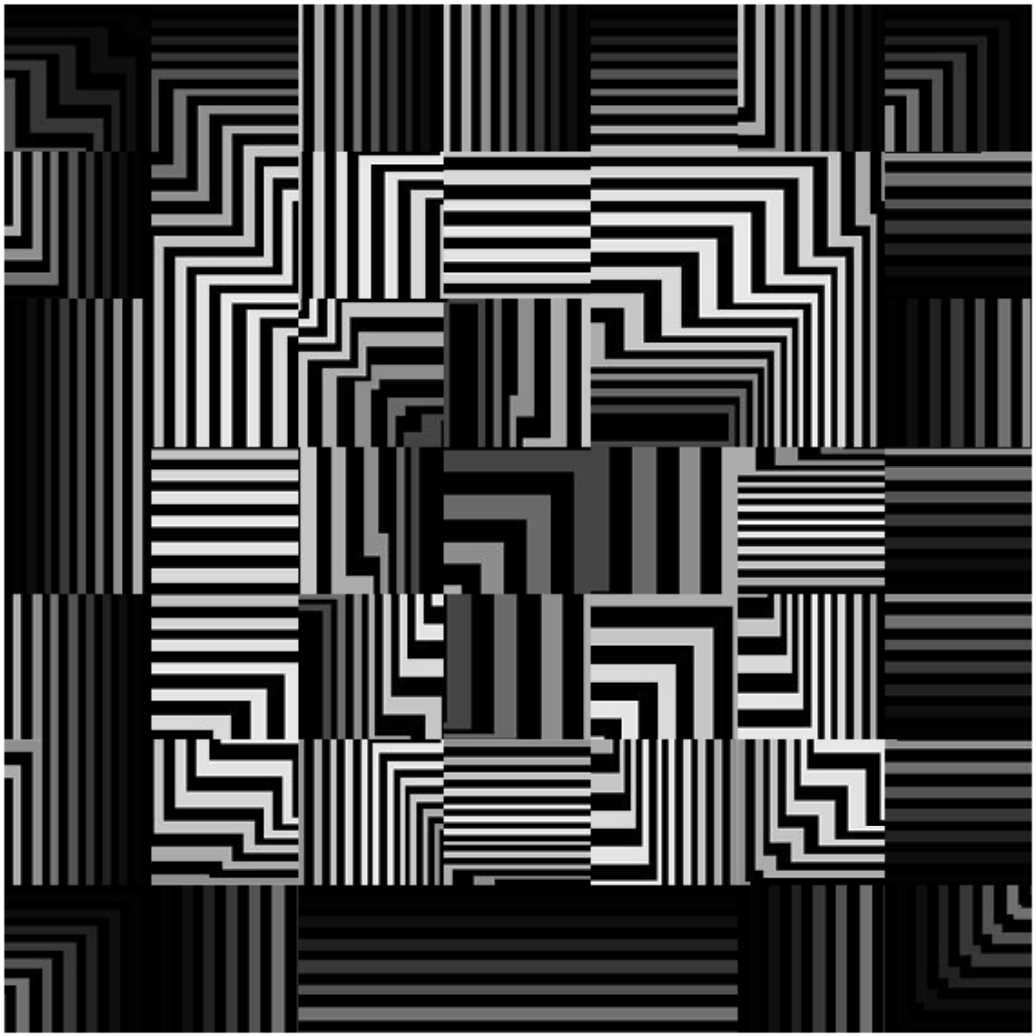
sekvencijalno generiranje slika (a)

1) generiranje konstruktivne slike 1 pomoću *algoritma 1*
2) generiranje slike 2 iz slike 1 pomoću *algoritma 2*

1) generiranje konstruktivne slike 1 pomoću *algoritma 1* (parametarske jednadžbe)



2) generiranje slike 2 iz slike 1 pomoću *algoritma 2* (fragmentacija slike)



V. Čerić, Labyrinth 7, 2011

sekvencijalno generiranje slika (b)

1) generiranje slike 1 (pomoćna slika) iz slike 0 (obično fotografija) pomoću *algoritma 1*

2) generiranje slike 2 (konačna slika) iz slike 1 pomoću *algoritma 2*

fotografija (slika 0)

završna slika (slika 2)

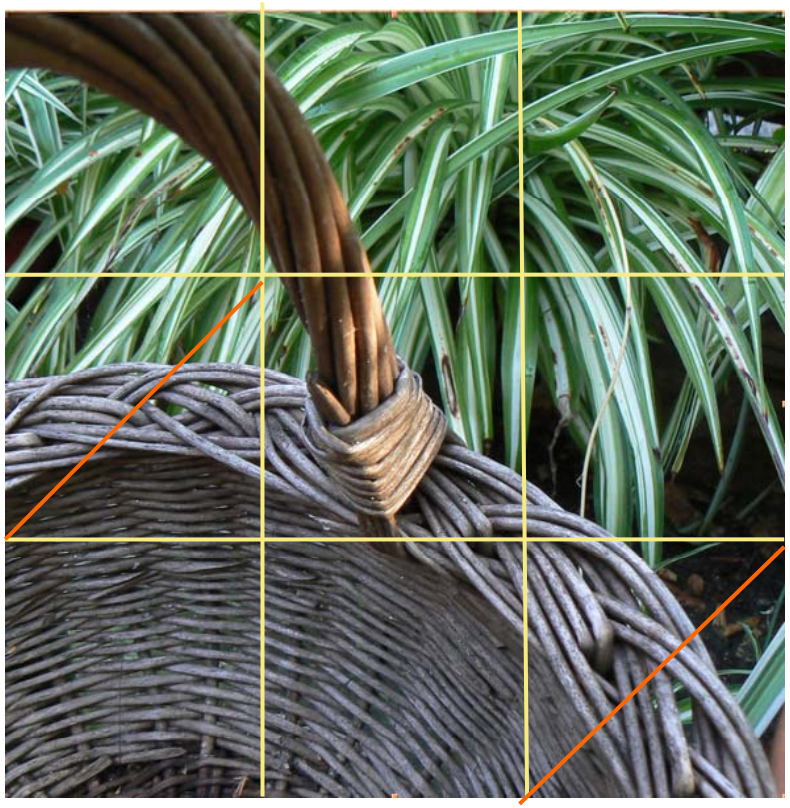
transformacija se izvodi u dvije faze:

1) fotografija se linearnom transformacijom pretvara u linearne "pruge"
2) dobivena slika se nelinearnom transformacijom pretvara u željenu nelinearnu sliku

1) linearna transformacija: (po sva 3x3 polja slike)

```
Do[ imgpart2[[ ii, jj ]] = imgpart1[[  jj, jj ]] ,
                                {ii, 1, ploc}, {jj, 1, ploc}  ],
```

pikseli fotografije (**imgpart1**) i linearne slike stvorene transformacijom fotografije (**imgpart2**)



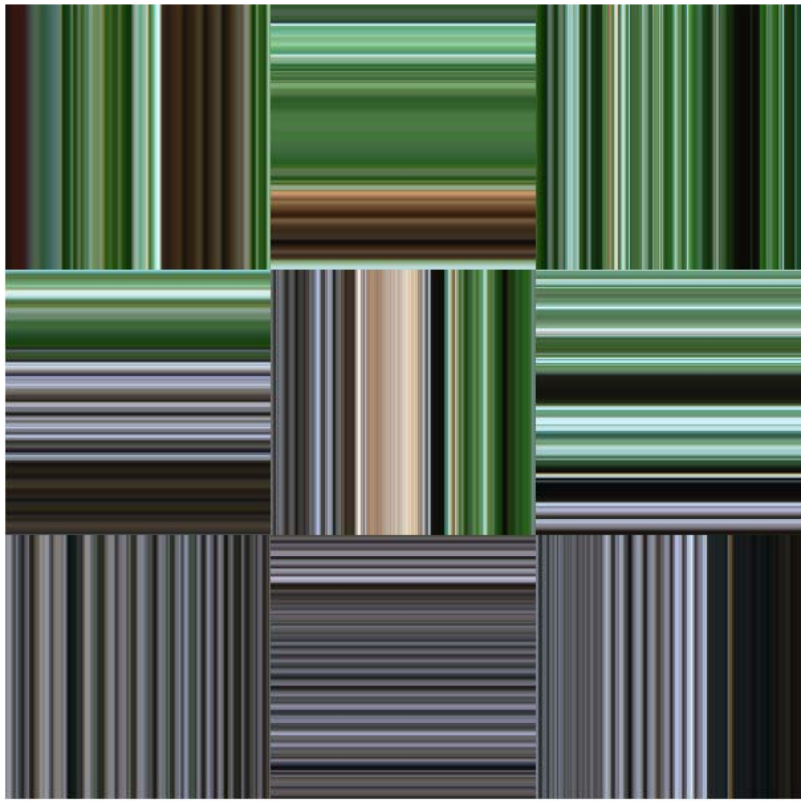
2) nelinearna transformacija: (po sva 3x3 polja slike)

```
Do[ imgpart[[ ii, jj ]] = imgpart2[[  jj, jj ]] ,
                                {ii, 1, ploc}, {jj, 1, ploc}  ],
```

pikseli završne nelinearne slike stvorene transformacijom linearne slike (**imgpart**) uzimaju vrijednost piksela linearne slike stvorene transformacijom fotografije (**imgpart2**)

```
Do[
  imgpart[[  ii, jj ]] =
  imgpart[[ IntegerPart[(ii*(ploc - jj + 1))^0.5],
                IntegerPart[((ploc - ii + 1)*jj)^0.5] ]] ,
                                {ii, 1, ploc}, {jj, 1, ploc}  ],
```

pikseli završne nelinearne slike (**imgpart**) transfomiraju se koristeći vlastite vrijednosti iz drugih točaka te iste slike



V. Čerić, *Cantabile 4*, 2010

algoritamsko"miješanje" boja piksela dvaju slika

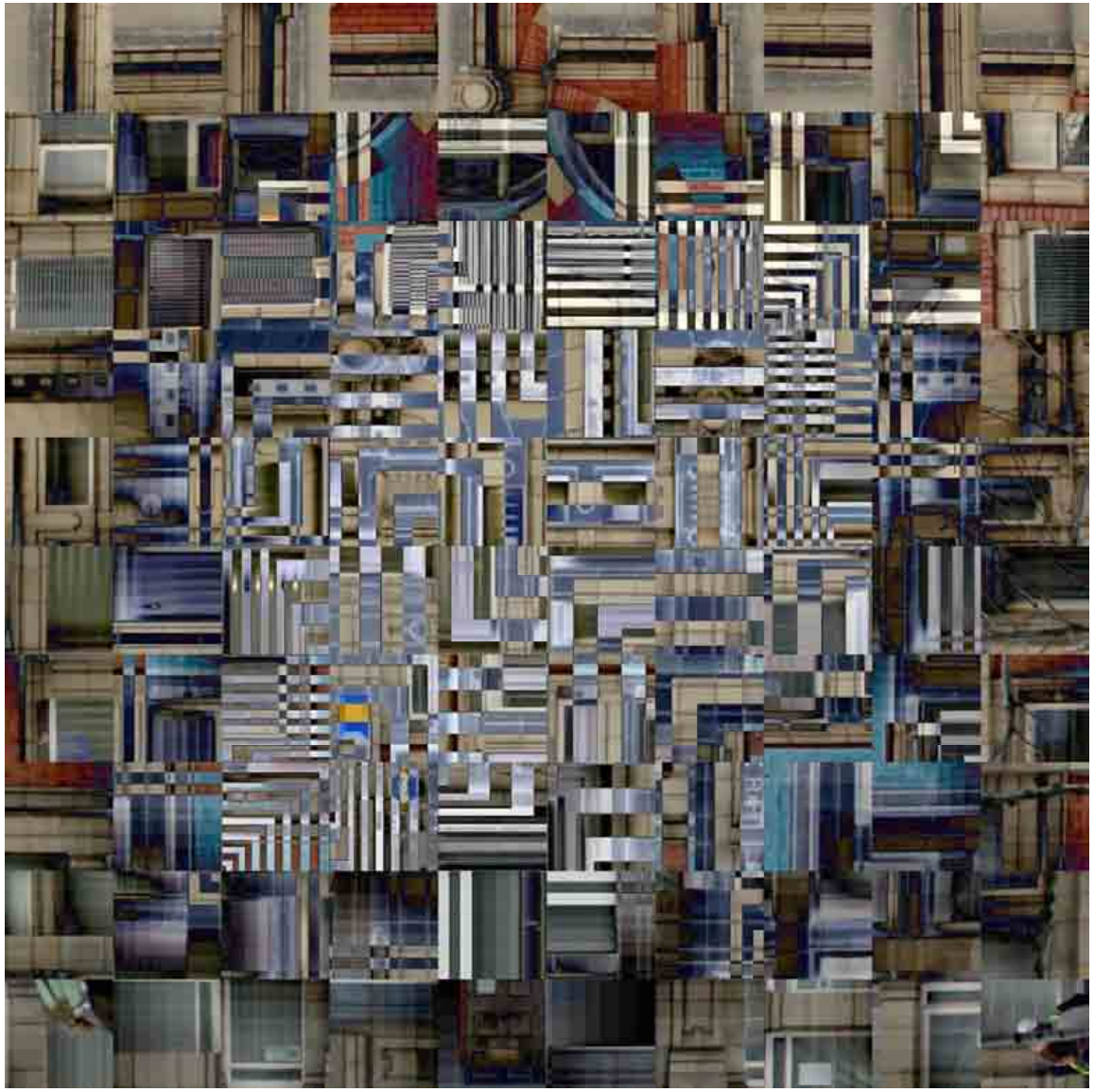
- to je još jedna od staza prema kompleksnosti -

"miješaju" se odgovarajući pikseli dvaju slika koji se nalaze na istom mjestu (x, y) u mreži piksela, npr.:

```
Do[   imgpart[[ ii, jj ]] =
        Abs[imgpart1[[ ii, jj ]] - imgpart2[[ ii, jj ]]]^c,
              {ii, 1, ploc}, {jj, 1, ploc} ],
```

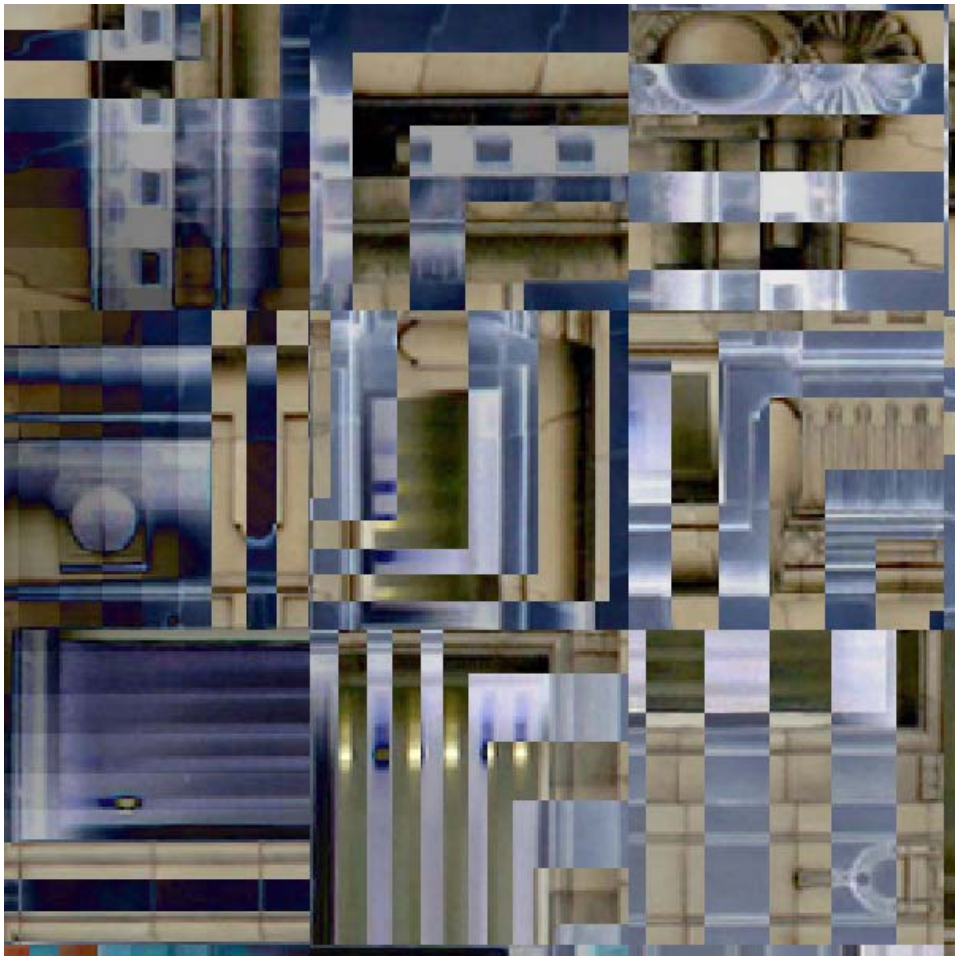
imgpart1 i imgpart2 su matrice piksela dvaju ulaznih slika (fotografija)
imgpart je rezultantna matrica piksela slike s "miješanim" bojama

fragmentirana fotografija + geometrijski motiv

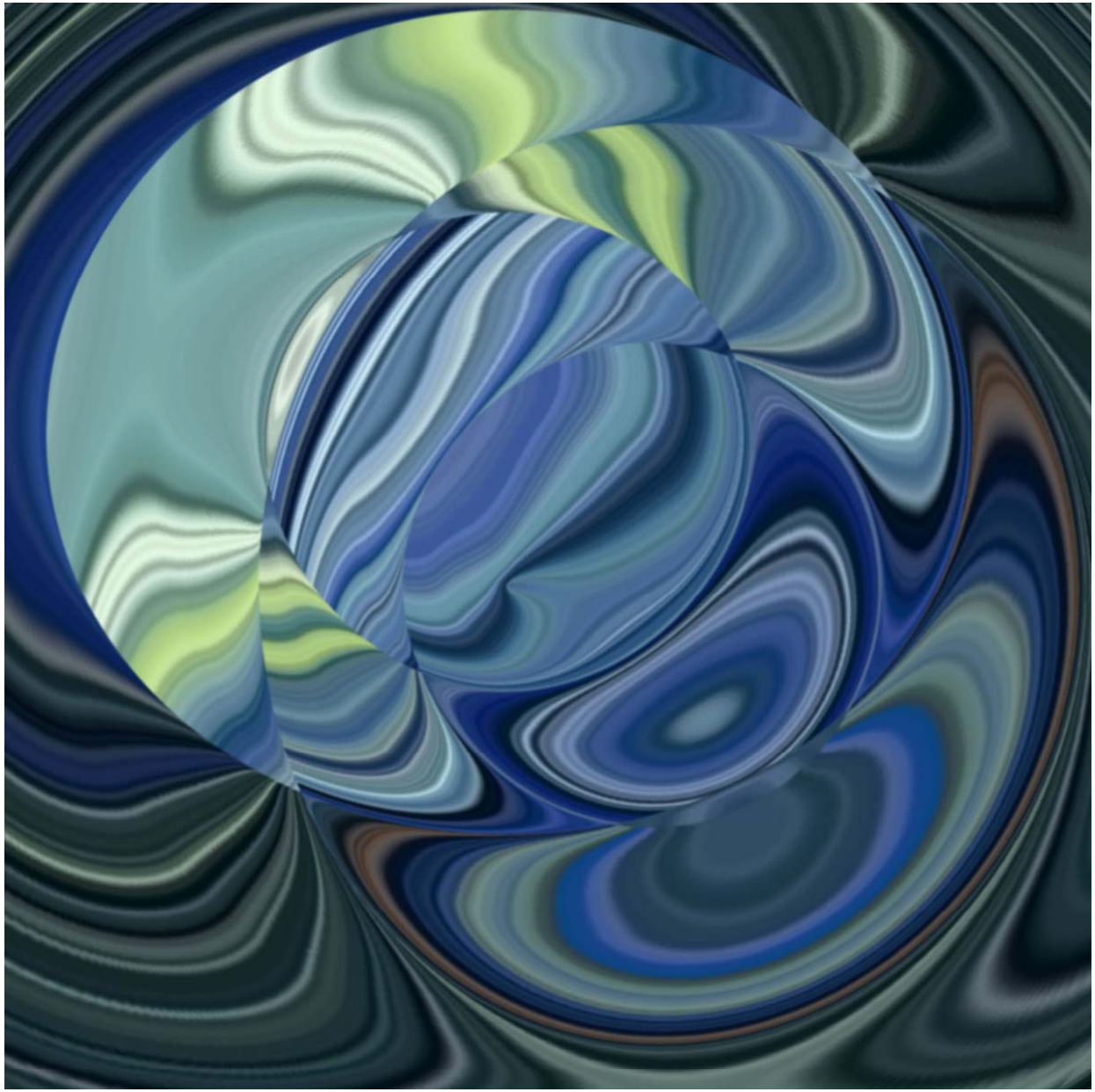


V. Čerić, *Fusion 1*, 2013

detalj



V. Čerić, *Amulets x,* u pripremi



dodatni primjeri nelinearne obrade slike

V. Čerić, *Metamorphosis x*, u pripremi

animacija

postepena promjena parametara algoritamske slike u vremenu

Tipovi animacija:

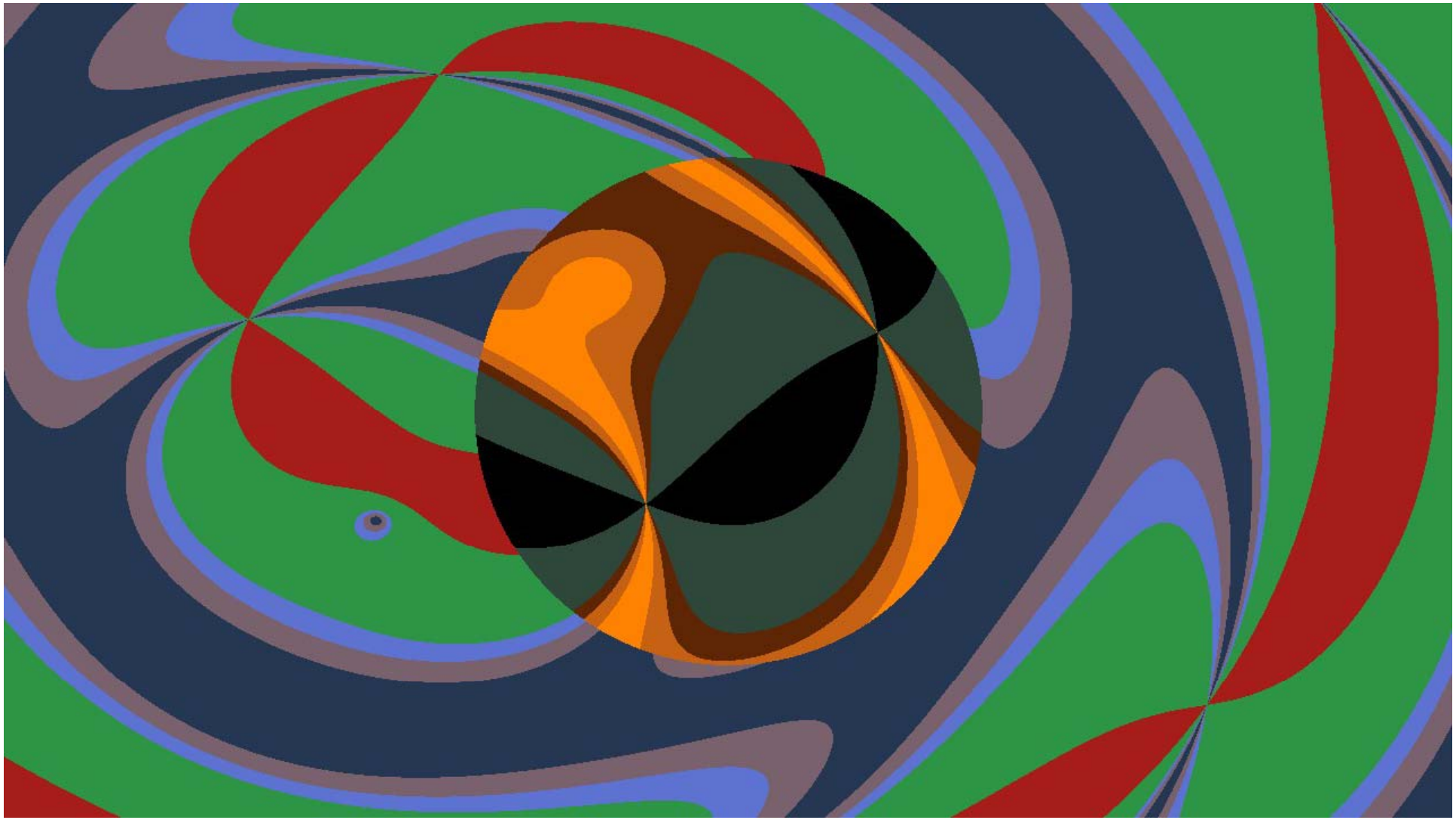
- geometrijska animacija (trokuti, kvadrati, ... pulsation)
- matematičko modeliranje (dance)
- animacija obrade fotografija (i "let iznad fotografija")
- apstraktni video (obrada figurativnih video radova)

- matematičko modeliranje

primjer: dance 1

uzastopne sličice koje čine animaciju

(programski jezik Processing)



Animacija ciklusa *dance* funkcionira na sljedeći način:

- U svakom koraku animacije mijenja se vrijednost funkcije dd(x,y) (oblik terena, veličina tlaka i sl.) u svakoj točki površine
- Time dolazi i do stalne promjene položaja izohipsa (koje su definirane fiksnim veličinama visine terena, veličine tlaka ili sl.), što rezultira pomicanjem područja obojanih različitim bojama
- To pomicanje područja obojanog različitim bojama pruža osjećaj gibanja na plohi slike

Osnovni elementi algoritma

```
for (int y=0; y<720; y+=1)
  {
    for (int x=0; x<1280; x+=1)
    {

      if( sqrt(sq(x - 640) + sq(y - 360)) < 220 )  // odredjivanje podrucja

      // Unutar centralnog kruga
      --------------------------
      {     // funkcija dd
     float dd = 350*abs(tan(c11*sqrt(c12*sq(c13*x-c14-300*sin(fspeed*c15*millis()))
                  *abs(sin(c21*sqrt(c22*sq(c23*x-c24+200*sin(fspeed*c25*millis())+ ...

            // odredjivanje boje(RGB) za podrucje izmedju uzastopnih izohipsa
      if(dd>980)          fill( 0, 0, 0);
         else if(dd>210) fill(0.18*256, 0.28*256,  0.23*256);
         else if(dd>140) fill(0.37*256, 0.15*256,  0.02*256);
         else if(dd>70)  fill(0.78*256, 0.385*256, 0.08*256);
         else            fill( 1.0*256, 0.515*256, 0.0*256);
     }

      // Izvan centralnog kruga
      --------------------------
      else
      {
       ...
```

prikazan samo dio funkcije dd(x,y)

(programski jezik Processing)

```
float dd = 350*abs(tan(c11*sqrt(c12*sq(c13*x-c14-300*sin(fspeed*c15*millis()))
                                +c16*sq(c17*y-400+100*sin(fspeed*c18*millis()))))
                  *abs(sin(c21*sqrt(c22*sq(c23*x-c24+200*sin(fspeed*c25*millis())+150)
                                +c26*sq(c27*y-400-300*sin(fspeed*c28*millis())-250))));
```

$$dd = 350 \cdot \mathrm{abs}(\tan(\mathrm{c11}\sqrt{A1 + A2}))\,\mathrm{abs}(\sin(\mathrm{c21}\sqrt{B1 + B2}))$$

$$A1 = \mathrm{c12} \cdot (\mathrm{c13} \cdot x - \mathrm{c14} - 300 \cdot \sin(\mathrm{fspeed} \cdot \mathrm{c15} \cdot millis()))^2$$

$$A2 = \mathrm{c16} \cdot (\mathrm{c17} \cdot y - 400 + 100 \cdot \sin(\mathrm{fspeed} \cdot \mathrm{c18} \cdot millis()))^2$$

$$B1 = \mathrm{c22} \cdot (\mathrm{c23} \cdot x - \mathrm{c24} + 200 \cdot \sin(\mathrm{fspeed} \cdot \mathrm{c25} \cdot millis()) + 150)^2$$

$$B2 = \mathrm{c26} \cdot (\mathrm{c27} \cdot y - 400 - 300 \cdot \sin(\mathrm{fspeed} \cdot \mathrm{c28} \cdot millis()) - 250)^2$$

Za područje izvan centralnog kruga:

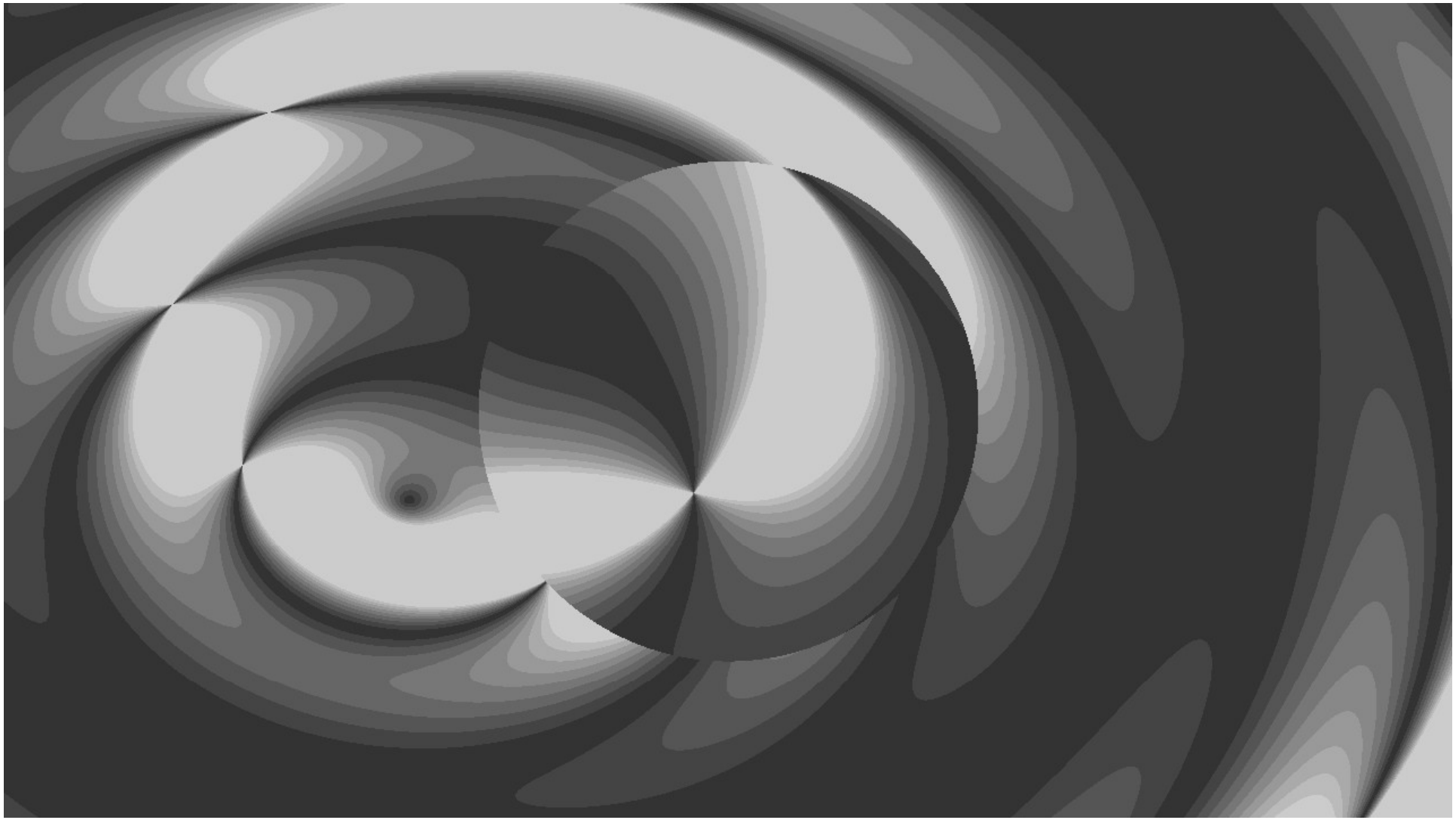
- funkcija *dd* se razlikuje od one za područje unutar centralnog kruga
- broj i vrijednosti izohipsa su jednake onima za područje unutar centralnog kruga
- odabrane boje se razlikuju od onih za područje unutar centralnog kruga

Kod drugih animacija iz ciklusa *dance* koriste se drugačiji izbori tih elemenata

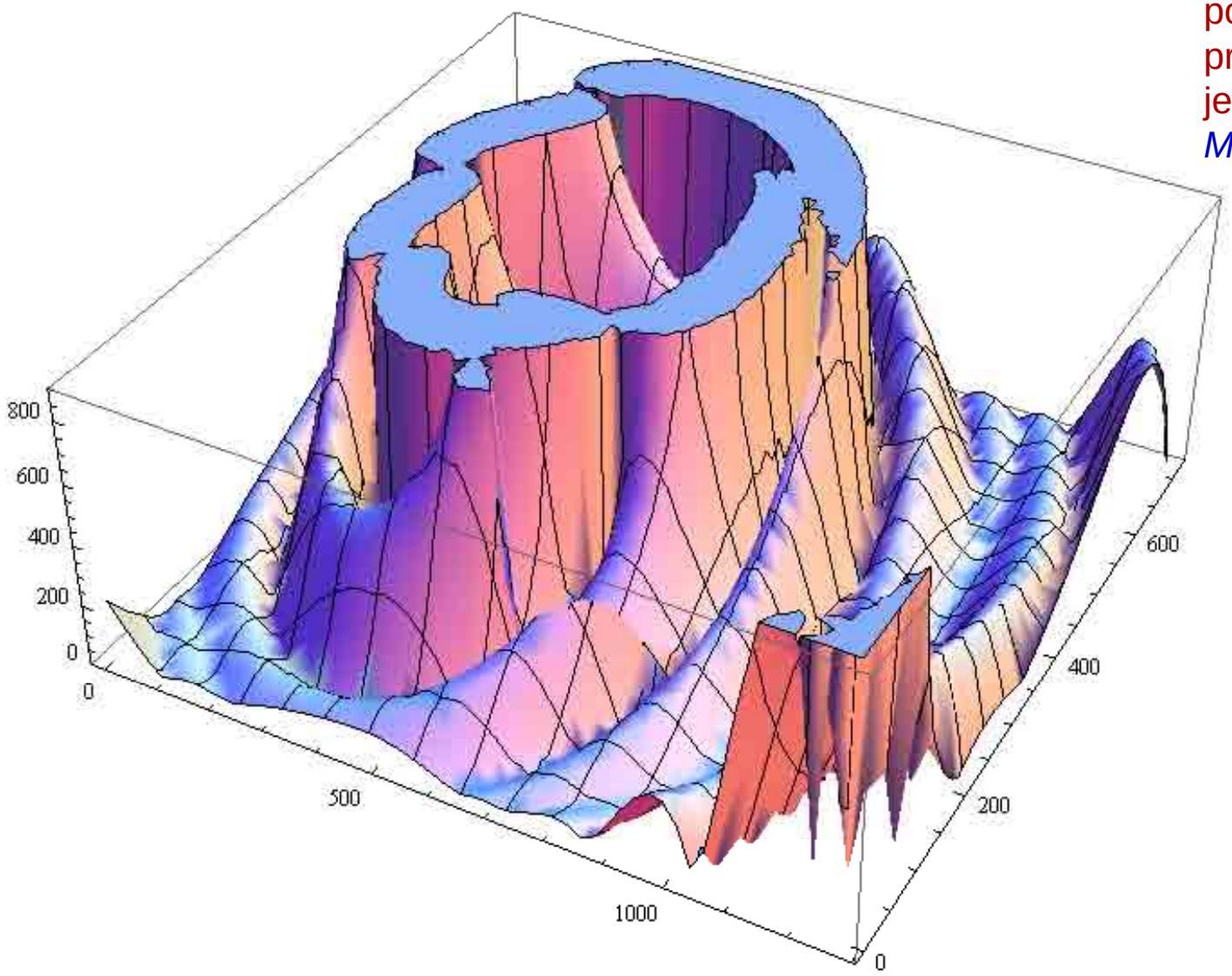
Kako bi se bolje vidjelo kakav je oblik funkcije dd(x,y)
napravljena je animacija

s monokromatskim bojama koje rastućim redosljedom idu od
* crne - najmanja vrijednost funkcije do
* bijele - najviša vrijednost funkcije

i s gušćim izohipsama,

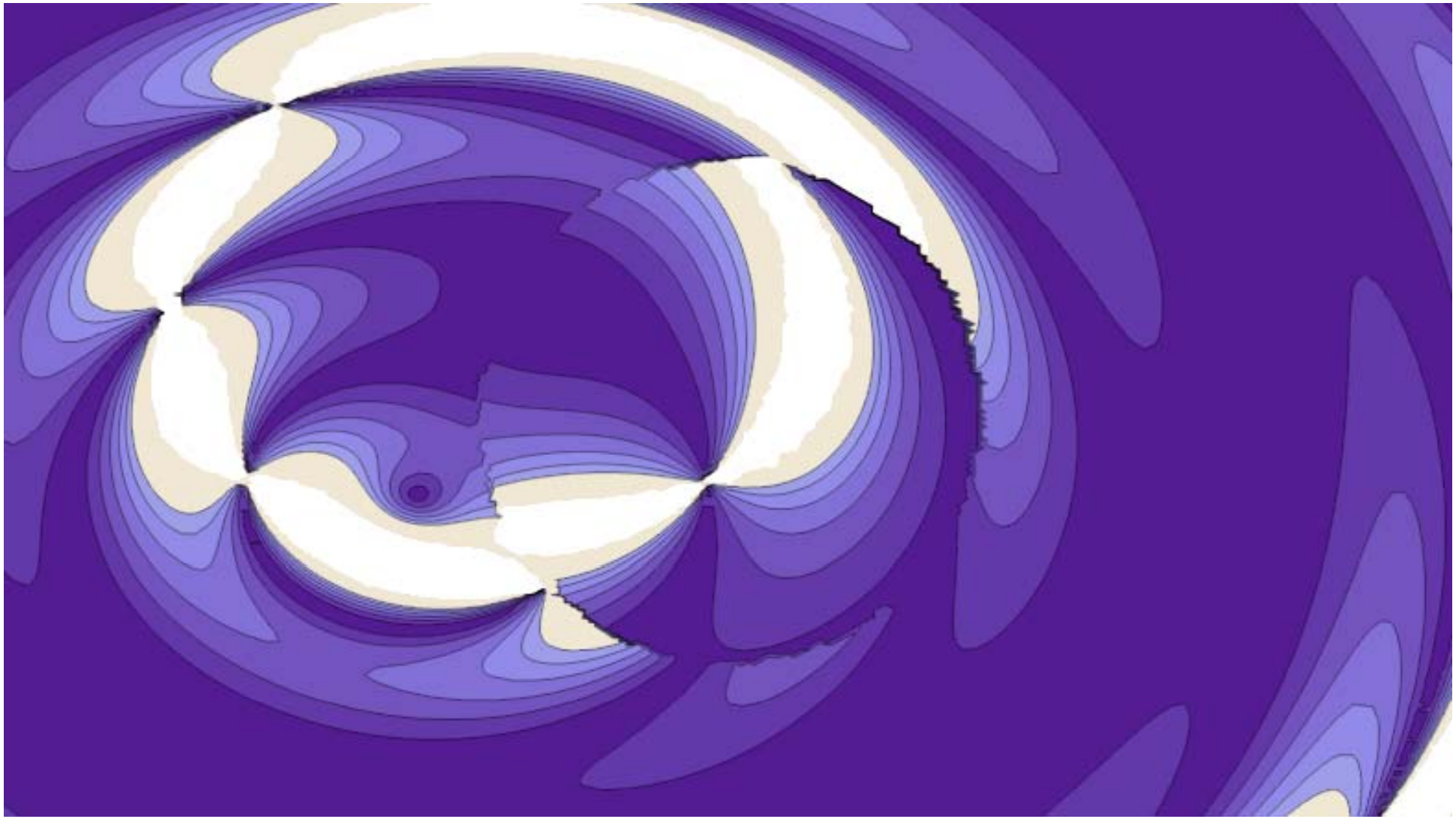


3D prikaz pomoću programskog jezika *Mathematica*



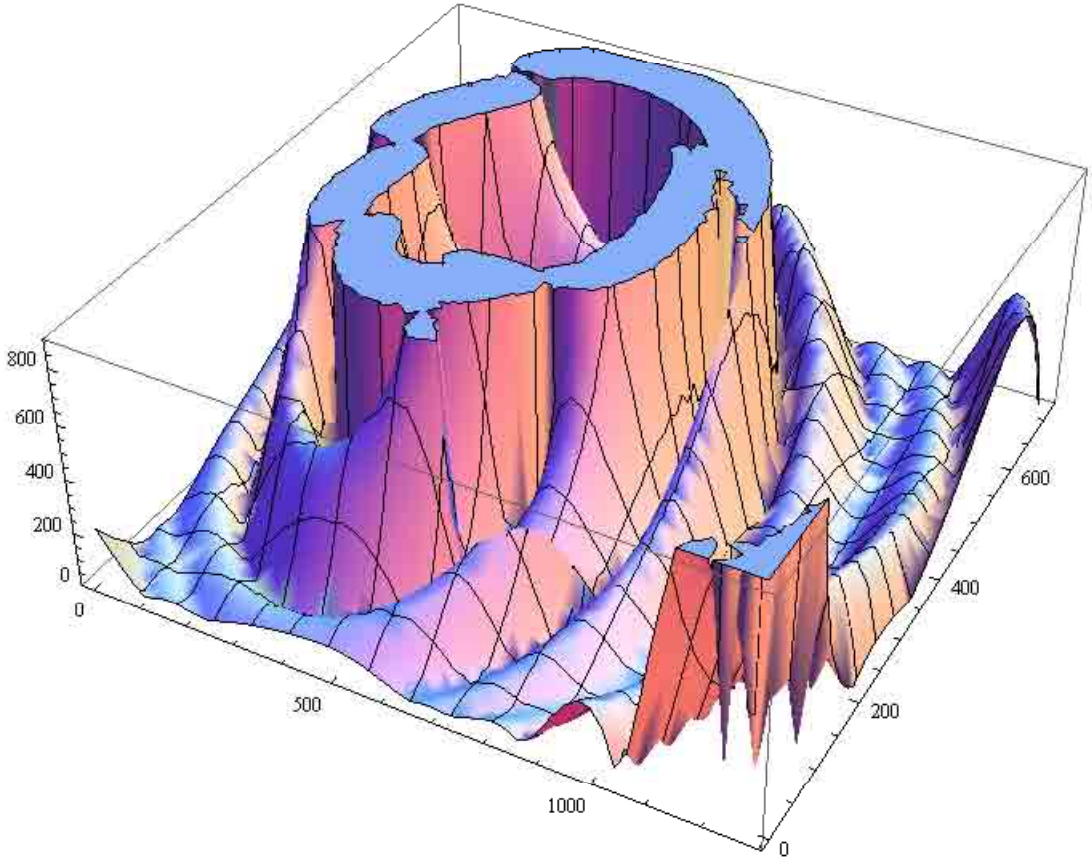
Oblik vanjskog dijela funkcije dd u 3DIM
(na početku animacije)

CountourPlot - progr. jezik *Mathematica*

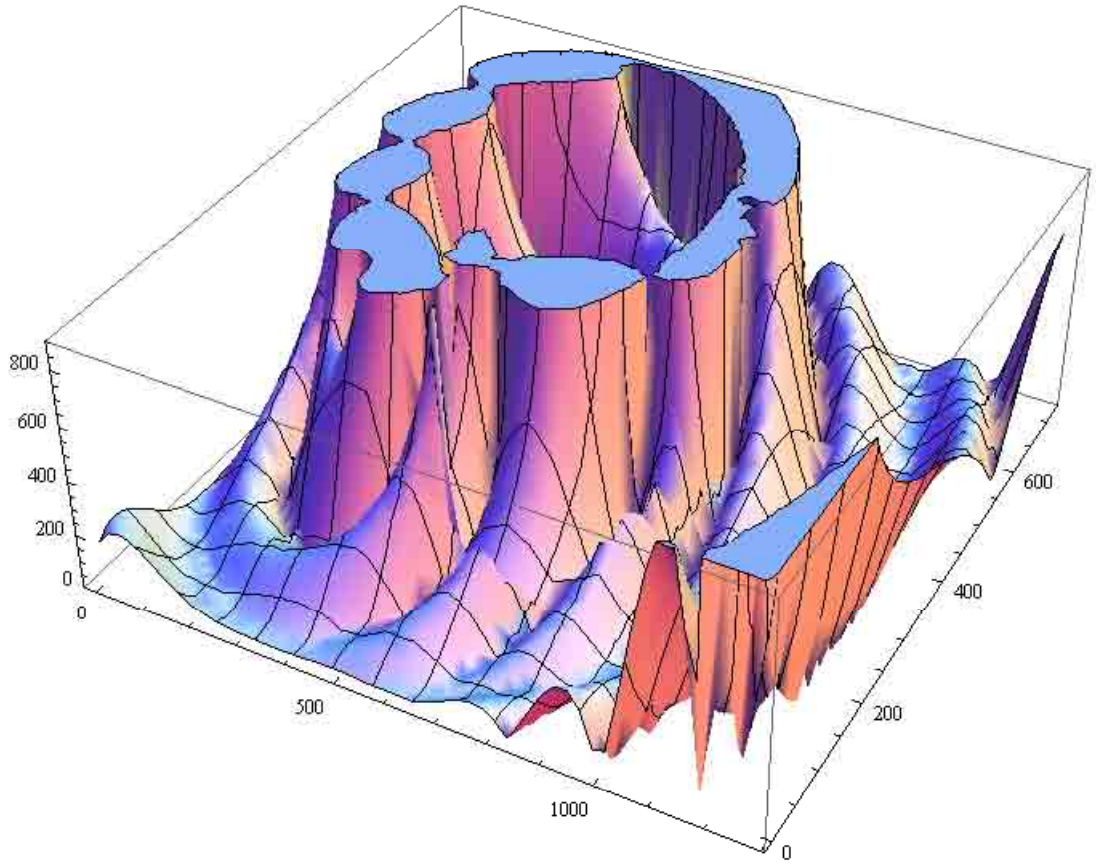


Izohipse
(na početku animacije)

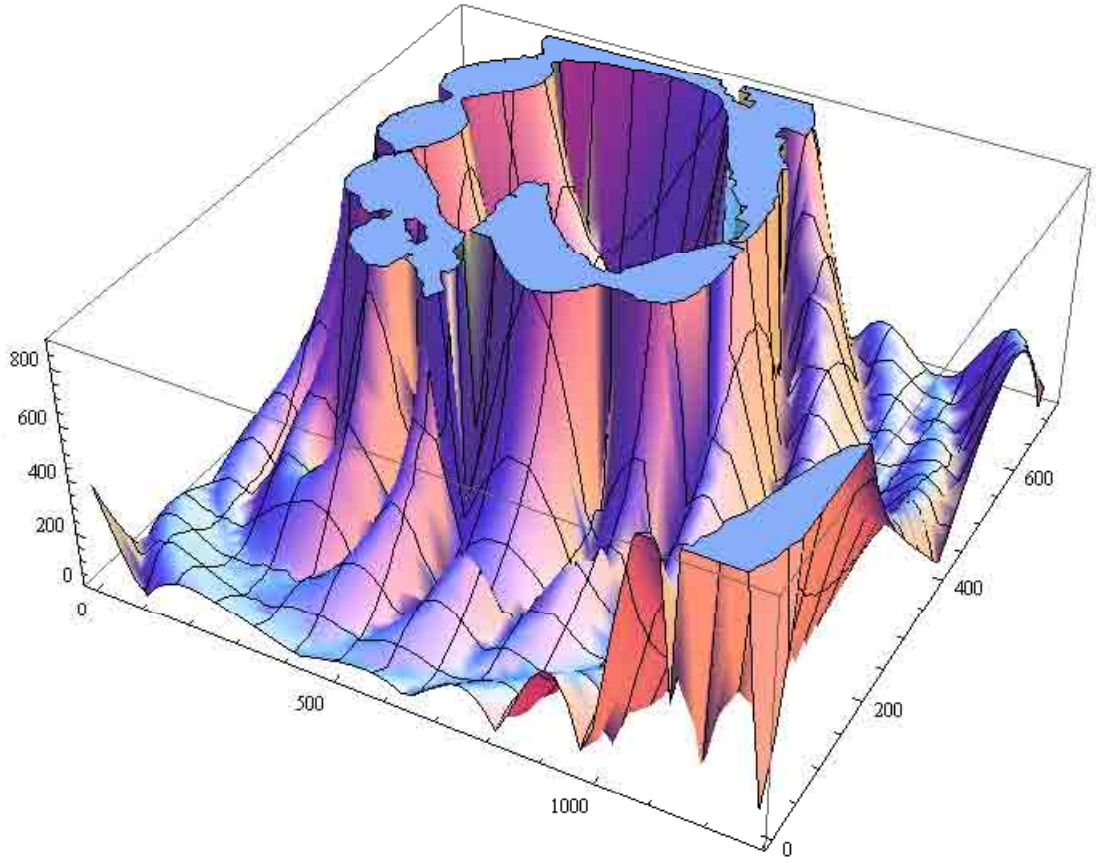
"animacija"
u nekoliko
sličica



Oblik vanjskog dijela funkcije dd u 3DIM
(na početku animacije - 1)

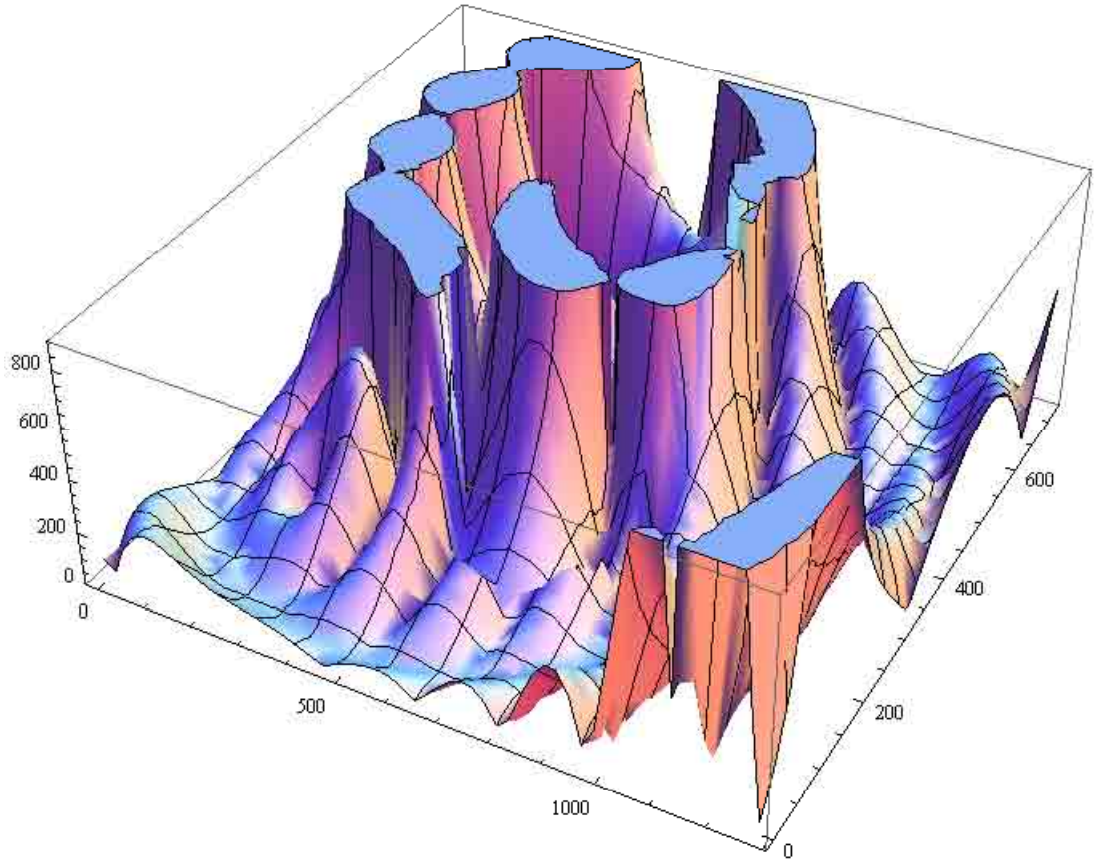


Oblik vanjskog dijela funkcije dd u 3DIM
(2)



Oblik vanjskog dijela funkcije dd u 3DIM

(na kraju ovog prikaza animacije - 3)



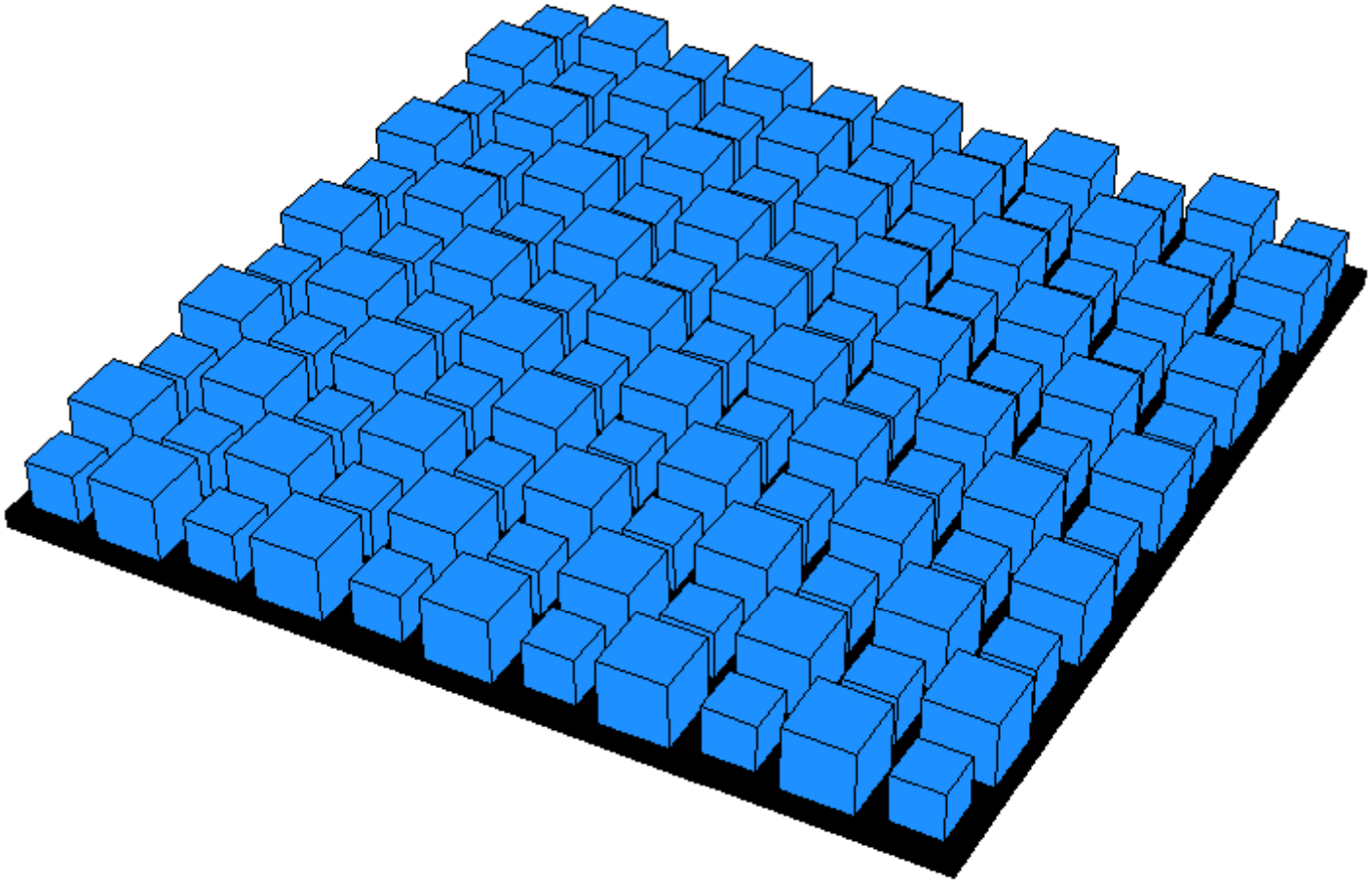
Oblik vanjskog dijela funkcije dd u 3DIM

(4)

- skulpture

  pripremljene algoritamskim pristupom

V. Čerić, *Matrix*, drvo, 2006



Studija za skulpturu *Matrix*

V. Čerić,
Tvrđava,
drvo, 2010

- primjene u dizajnu

  na temelju mojih algoritamskih grafika

logo za *DiM consulting*,
Budapest, Hungary

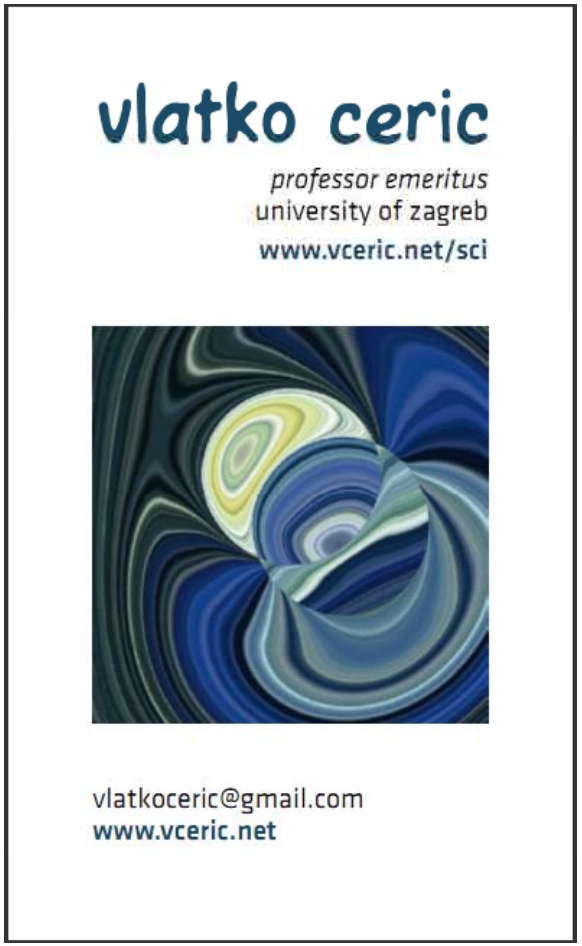
autor: Ranko Čerić, 2005

logo za *CGA Investment Partners*,
Amstelhoek, The Netherlands

autori: *The Dutch Design Studio*,
2012

moja posjetnica

autor: Ranko Čerić, 2013



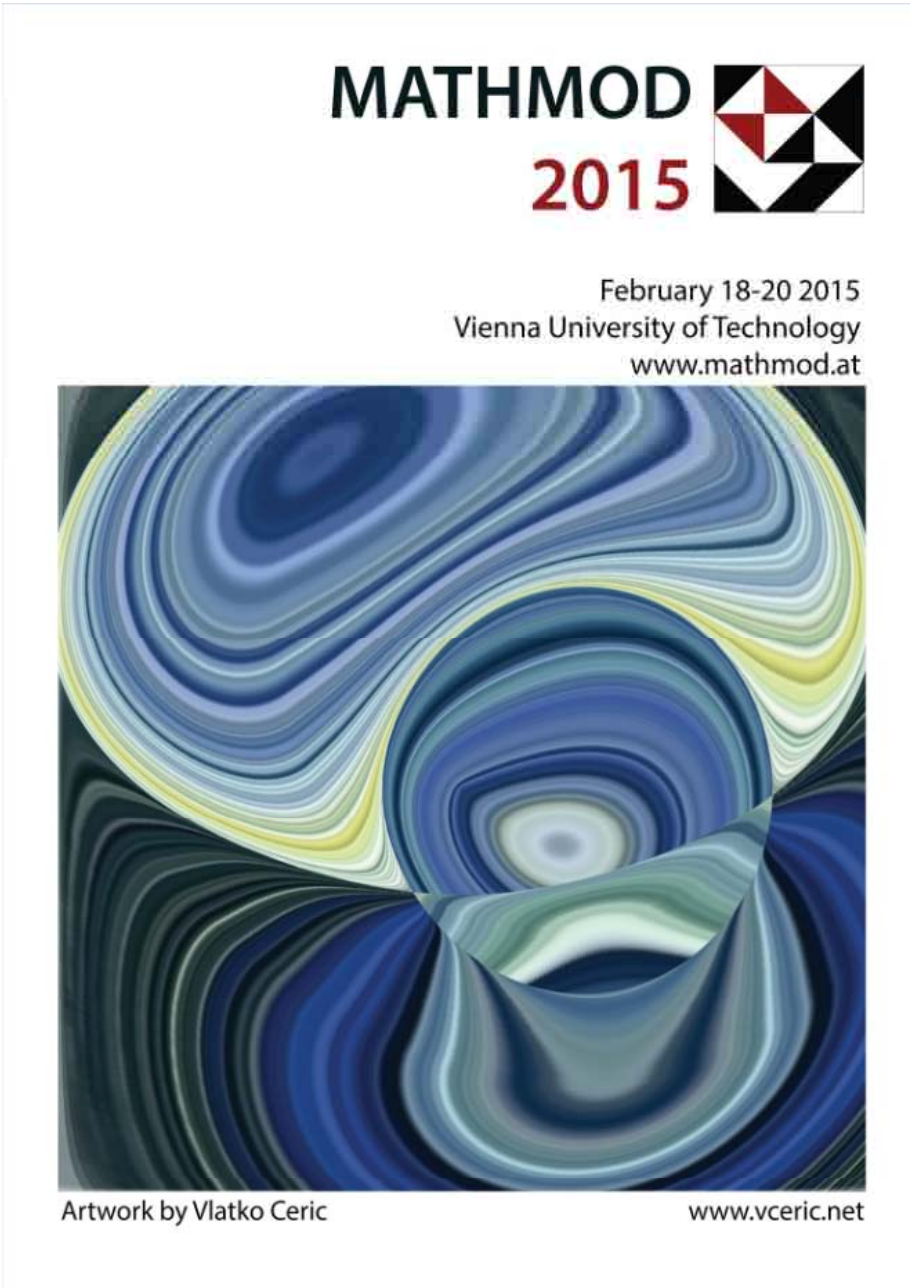
folder za *North Carolina State University (NCSU)*, Department of Computer Science

autori: NCSU design team, 2009

Konferencijski poster

8th Vienna conference on mathematical modelling "MATHMOD 2015", Vienna, Austria

Autor: Tamara Vobruba, 2015



Časopis "Simulation Notes Europe" 2011

Autor: Anna Mathe, 2011

Časopis "Simulation Notes Europe" 2013

Autor: Tamara Vobruba

umjesto zaključka

- stvaranje vizualnih radova u algoritamskoj umjetnosti nije tako jednostavno i "pravocrtno" kao što to (možda) izgleda gledajući gotove postupke i njihove rezultate
- osim dobrih ideja potrebno je jako mnogo eksperimentiranja, pa se tako u pravilu ispituje velik broj varijanti pojedinog algoritama
- u stvarnosti često dolazi do grešaka ali ponekad i do slučajnih otkrića koja daju dobre rezultate i iniciraju novi smjer rada

www.vceric.net

vlatkoceric@gmail.com