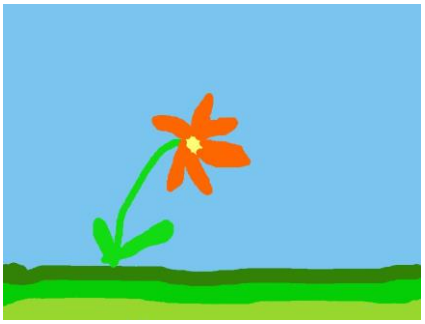
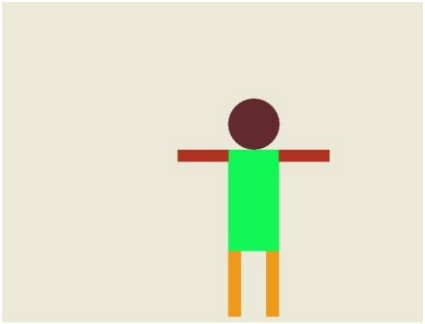
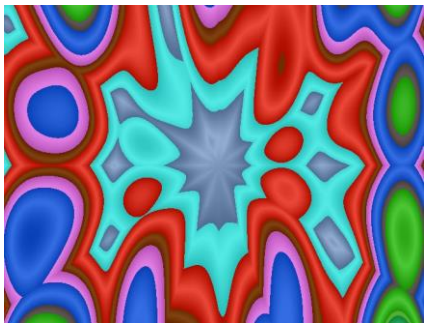
BOGDAN SOBAN - MATEMATIKA V BARVAH (bogdan@soban-art.com www.soban-art.com)
Vloga matematike pri razvoju programov za generiranje slik

Digitalizacija nezadržno vstopa na vsa področja človekovega delovanja, tudi na tista, ki so že po sami vsebini v domeni človekovega ustvarjanja. V mislih imam humanistične znanosti in predvsem področje umetnosti. Povezanost med umetnostjo in znanostjo, še posebej z matematiko, je zgodovinsko dokazana. S pojavom digitalne umetnosti je matematika pridobila še dodatno vlogo, saj se neposredno vključuje v sam proces nastajanja umetniških del. Tudi preprost programski algoritem brez pomoči matematike ne more narisati niti najenostavnejših elementov slike. Govora bo torej o vlogi matematike pri razvoju programske opreme za ustvarjanje digitalnih slik.

Po enostavni definiciji je digitalna slika grafična podoba, ki je nastala po digitalnem postopku, je shranjena in se distribuira v digitalni obliki. Predstavil bom tri primere, ki nazorno prikazujejo nastanek digitalne slike. Preprost primer je uporaba programske opreme, ki nadomesti slikarjev čopič, paleto z barvami in platno. Namesto tega se uporablja miška, digitalne barvne palete in monitor. Avtor tako nastale slike, ki se ročno riše po monitorju, je izključno uporabnik programa, saj računalnik ni ničesar prispeval h kreaciji – ročni pristop (slika 1). V naslednjem primeru lahko uporabimo namensko programsko opremo, ki ponudi zelo veliko variant vnaprej zamišljenega motiva – objektni pristop (slika 2). V tretjem primeru (slika 3) pa je nemogoče predvideti, kakšna slika bo nastala, saj je vse likovne elemente slike naključno izbral računalnik oziroma program – algoritemski pristop. V nadaljevanju se bom omejil samo na tovrstne likovne podobe, ki jim običajno rečemo kar algoritemske slike (generative art, algorithmic art, art from code, random art etc).


Slika 1


Slika 2


Slika 3

Predno pristopim do predstavitve konkretnih načinov risanja, bi navedel nekaj osnovnih izhodišč (infrastruktura), ki so ključnega pomena za delovanje tovrstnih programov in večina njih je več ali manj povezana z matematiko. Najprej je seveda tu programski jezik Visual Basic, ki je po svoji strukturi in zgradbi zelo primeren za risanje. Dalje bi omenil koordinatne sisteme, s katerimi obvladujem objekt »slika« na katerega rišem: kartezični koordinatni sistem, polarni koordinatni sistem, kompleksna ravnina in koordinatni sistem programskega jezika, ki ima izhodišče v levem zgornjem kotu. Vsaka točka (pixel) na sliki se lahko opiše z vsemi temi štirimi sistemi. Uporaba generatorja naključnih števil je odločujočega pomena za doseganje nepredvidljivih rezultatov. Uporabljam v programski jezik vgrajen psevdo-generator naključnih števil, pri katerem v je v določitev prvega števila vključena vrednost časa zagona, kar zagotavlja relativno dobro naključnost. Risanje slike se izvaja po kolonah in vrsticah od leve proti desni. Na vsaki točki ravnine se izvede algoritem, ki določi barvo trenutne točke. Algoritem je v bistvu kompleksen matematični izraz z velikim številom spremenljivk. Uporabljam dve vrsti spremenljivk: eno so konstante za programski cikel, drugo pa vrednosti, ki jih generira sam proces. Na začetku vsakega programskega ciklusa definiram tako imenovano gensko kodo – to je množica naključnih vrednosti, ki jih določim z uporabo naključne funkcije. Med samim procesom pa se generira druga množica vrednosti, ki jih dobim iz stohastičnega gibanja točke v ozadju procesa risanja. Obe množici skupaj vstopata v izračun vrednosti matematičnega izraza za trenutno točko. Tako izračunana vrednost je osnova za določitev barve točke. Uporabljam tri-komponentni barvni sistem RGB (red, green, blue). Barva se določi ali na osnovi barvnega algoritma ali pa je izračunana vrednost lokacija za odvzem barve iz barvne palete. Za doseganje boljše barvne skladnosti slik običajno uporabljam barvne palete, ki se izrišejo na osnovi podatkov iz programa in jih program še

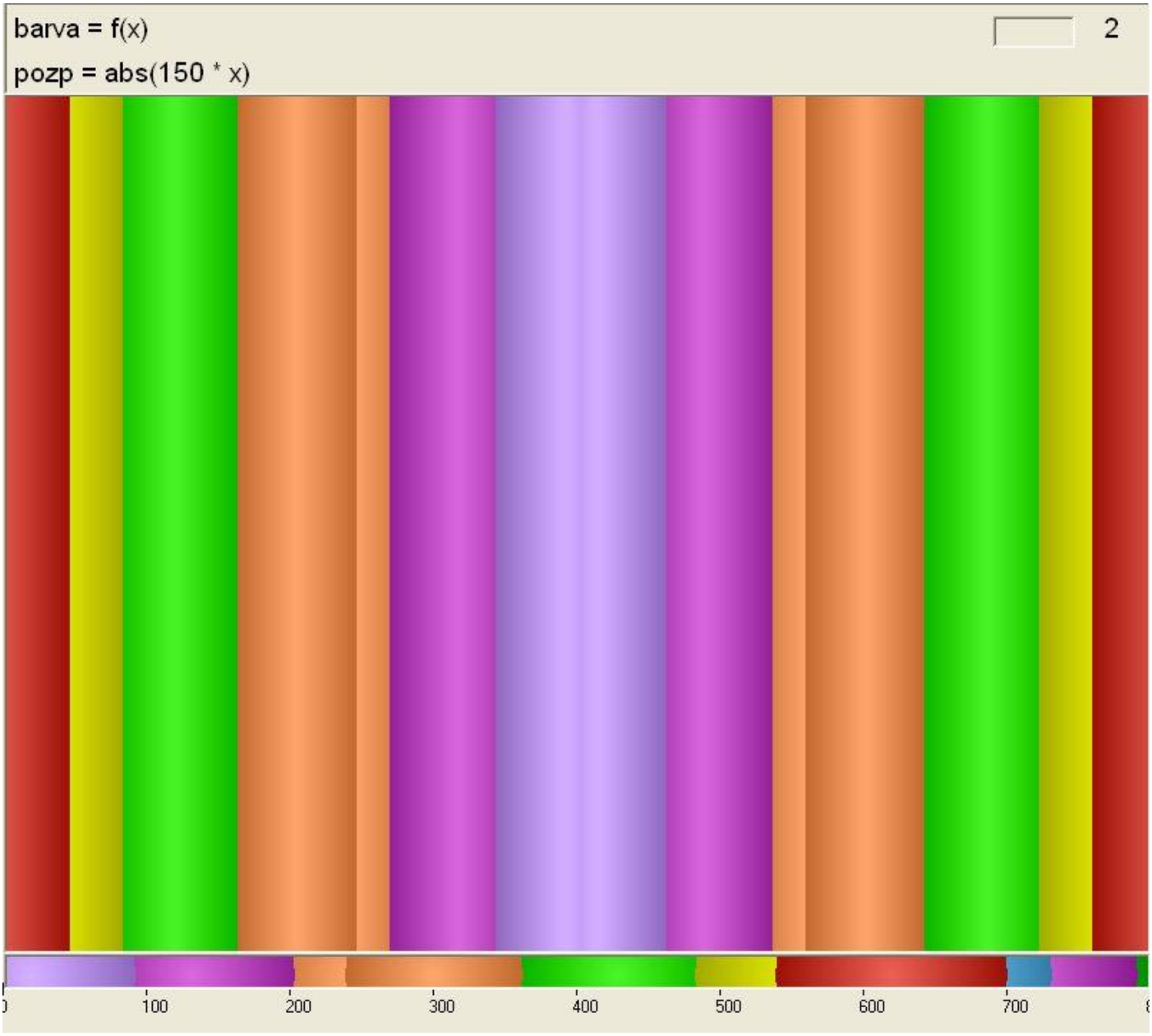
dodatno niansira, ko jih izrisuje.

V vseh zgoraj navedenih postopkih se uporabljajo različne matematične operacije in izračuni, ki jih na kratko strnem v naslednja področje: algebraični izrazi, ravninska geometrija, trigonometrične funkcije, iteracije in rekurzije, določanje naključnih števil v območju, krožna, spiralna in stohastična gibanja točke v ravnini, računanje s kompleksnimi števili, obvladovanje zaporedij in sestavljanje formul za potrebe različnih izračunov.

V nadaljevanju bom predstavil risanje slike v sistemu kartezičnih koordinat, v polarnih koordinatah in v kompleksni ravnini. Za boljše razumevanje bom uporabil enostavne algebraične izraze kot algoritme za določitev barve točke. V vseh primerih bom zajemal barve iz barvnih palet.

Risanje slike v kartezičnih koordinatah (x od -4 do +4, y od -3 do + 3)

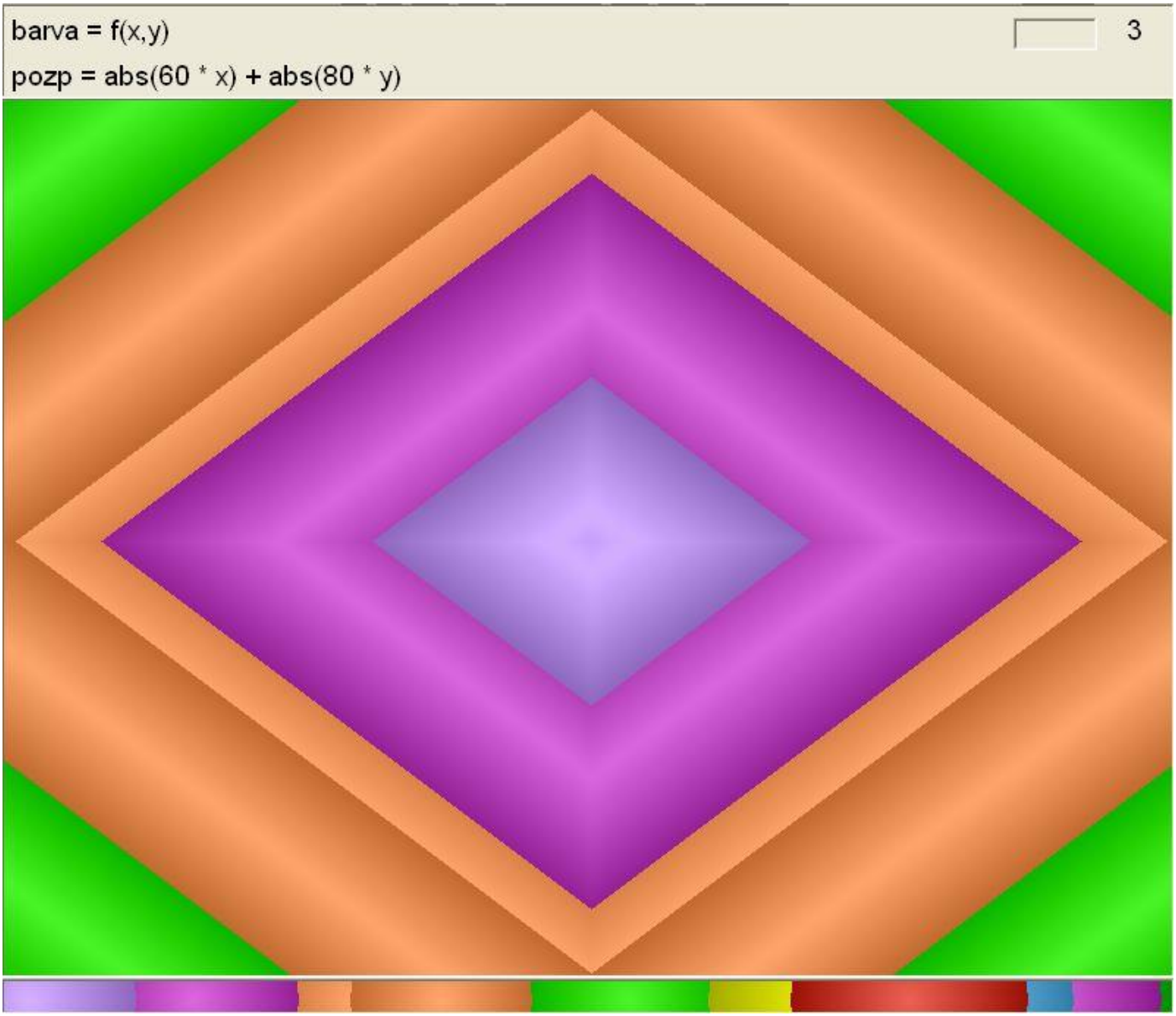
Prvi primer (slika 4):
barva = f(x)
pozp = abs(150 * x) pozicija za odvzem barve iz palete



Slika 4
za x = -4 je vrednost izraza pozp = 600, na paleti je to odtenek rdeče barve (glej skalo)

za x = 0 je pozp = 0, na paleti je to na samem začetku, torej svetlo viola
za x = 4 je pozp = 600 in smo spet na rdeči barvi.
Nastala slika je pravzaprav prerisana paleta od začetka do lokacij 600 in nazaj

Drugi primer (slika 5):
barva = f(x,y)
pozp = abs(60 * x) + abs(80 * y) pozicija za odvzem barve iz palete

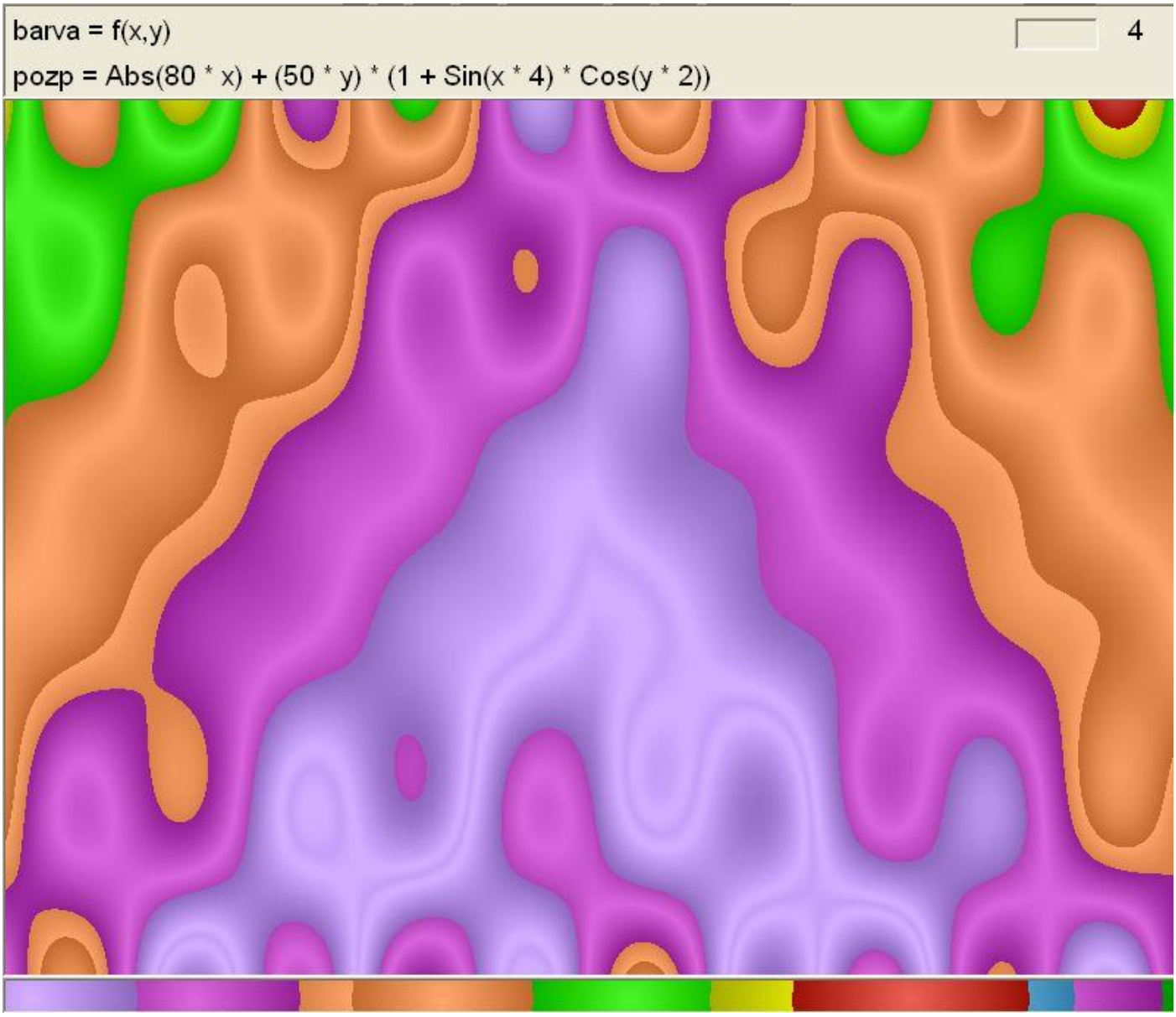


Slika 5

Za x = 0 in y = 0 je pozp = 0, na paleti je to na samem začetku – torej v središču slike je svetlo vijolična barva. Za x =4 in za y = 3 je pozp = 480, torej smo na zeleni barvi desno od sredine palete.
Na nastali sliki so barve iz palete razporejene dvodimenzionalno, kar pogojuje odvisnost od x in y.

To sta bila dva primera z izredno enostavnim algoritmom (matematičnim izrazom) za izračun barve točke, za katera je možno predvideti kakšna slika bo nastala. V naslednjem tretjem primeru uporabim malo bolj zapleten matematični izraz, kjer pa je že nemogoče natančno predvideti sliko.

Tretji primer (slika 6):
barva = f(x,y)
pozp = abs(80 * x) + (50 * y) * (1 + sin(x * 4) * cos(y * 2)) pozicija za odvzem barve iz palete

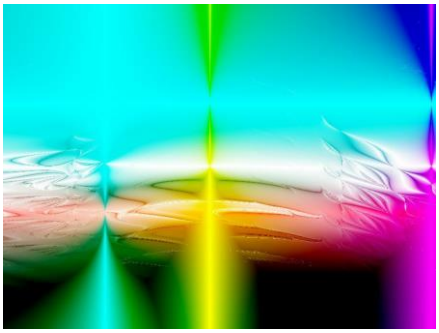


Slika 6

Na sliki 7 je nekaj primerov algebraičnih izrazov za izračun barve točke, vzetih iz konkretnega programa. Na slikah 8, 9 in 10 pa so primeri, generirani z omenjenim programom.

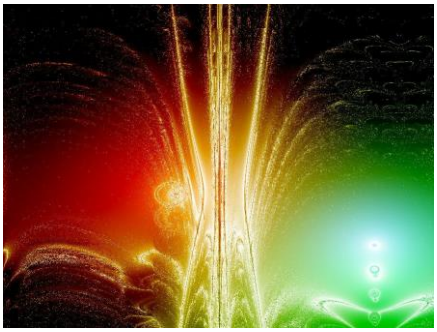
```
n01: ba = Log(Abs(ob67 + X)) * Log(Abs(rmin * Cos(ba + r5))) * Log(Abs(Y + Tan(rmax))): GoTo risi
n02: ba = r3 + Log(Abs(Y * Sin(Cos(Y * X)) + r4 * Cos(Sin(ba * r3)))): GoTo risi '4/10
n03: ba = r3 + Log(Abs(Y * Sin(Cos(ba * X)) + r4 * Cos(Sin(ba * r3)))): GoTo risi '4/10
n04: ba = X + Log(Abs(r4 * Sin(Y * r3) + r3 * Cos(ba * r4) * Log(Abs(Y)))): GoTo risi '4/7
n05: ba = X + Log(Abs(r4 * Sin(ba * r3) + r3 * Cos(ba * r4) * Log(Abs(Y)))): GoTo risi '4/7
n06: ba = Y + Log(Sqr(Abs(Sin(Cos(Tan(ba * Y))) + Cos(ba * Y) + Sin(X)))): GoTo risi '7/9
n07: ba = r3 + Y * Sin(Cos(Tan(r3))) + X * Cos(Sin(Tan(ba))): GoTo risi '4/3
n08: ba = X * Log(Abs(X + r4 * Sin(Y * ba))) + Log(Abs(r3 * Cos(r4) + Log(r4))): GoTo risi '4/15
n09: ba = X * Log(Abs(Sin(Cos(r3 + r4) + Cos(Sin(ba + Y))))): GoTo risi '6/11
n10: ba = Log(Abs(r4 + Sin(ba + X - Y) + r5 * Cos(ba + r4 * Y) - r3 * Sin(ba * X))) + X: GoTo risi '8/7
n11: ba = Log(Abs(r4 * Sin(r3 * ba + X - Y) + r5 * Cos(fi5 + r4 * ba + Y) - r3 * Sin(r4 + ba + X))) + X + Y + r3:
n12: ba = Y + Log(Abs(Sin(Cos(r3 + r4 + ba) + Cos(Sin(X + Y + ba))))): GoTo risi '6/10
n13: ba = Log(Abs(Y * Sin(Cos(Tan(r3 + ba))))) + Log(Abs(X * Cos(Sin(Tan(r4)))))): GoTo risi '7/14
n14: ba = Log(Abs(Y * Sin(Cos(Tan(r3 + ba))))) + Log(Abs(ba + X * Cos(Sin(Tan(ba)))))): GoTo risi '7/14
n15: ba = Y + Sin(Log(Abs(ba + X * r3))) * Sin(Cos(ba * Y)) + Log(Abs(ba + r4 * Y)) * Sin(Cos(Log(Abs(Tan(ba + r5)
```

Slika 7



Slika 8

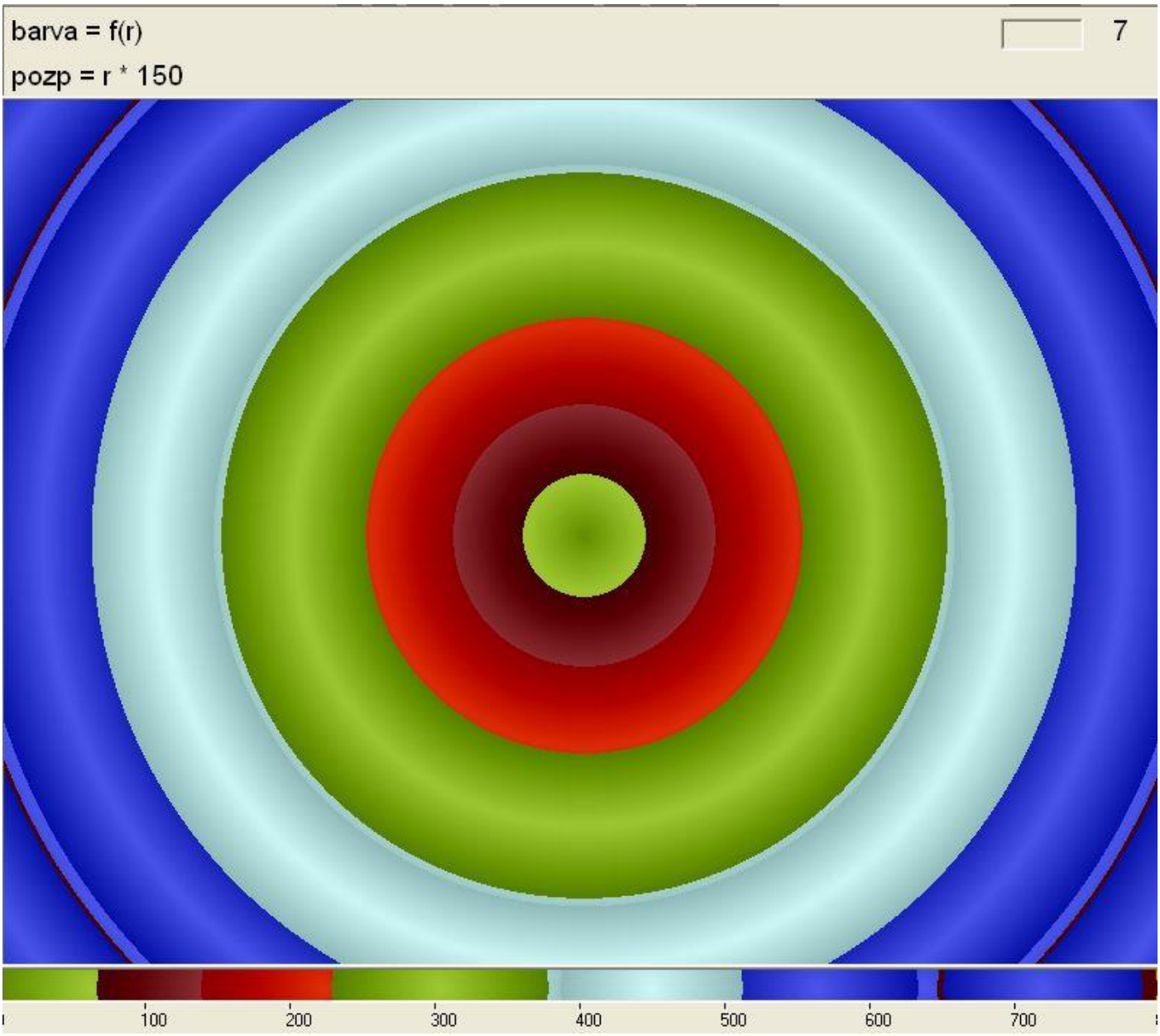
Slika 9



Slika 10

Risanje slike v polarnih koordinatah

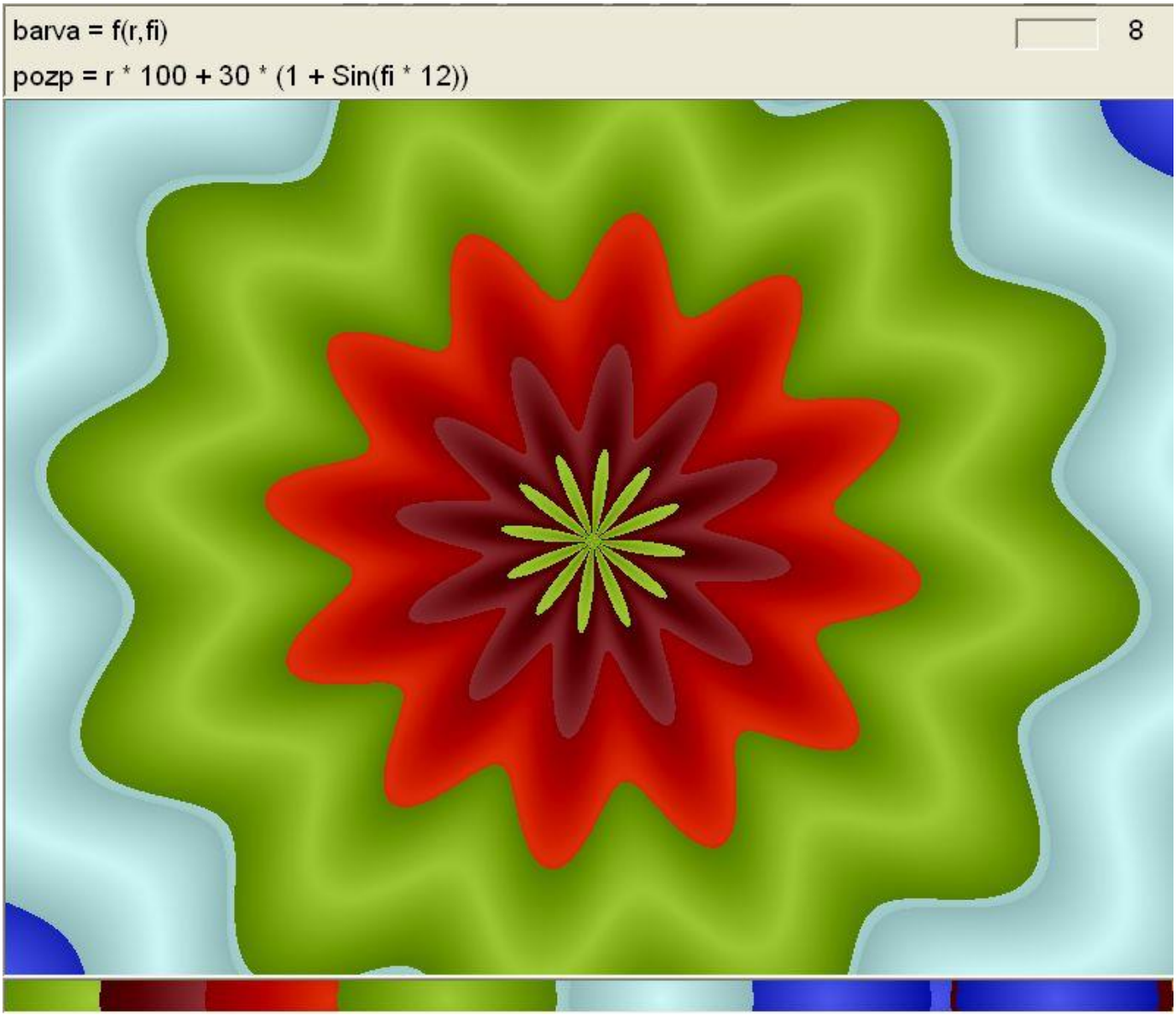
Prvi primer (slika 11):
barva = f(r)
pozp = r * 150 pozicija za odvzem barve iz palete



Slika 11

Pričakujemo koncentrične barvne pasove – preslikava palete
za r = 0 je pozp = 0 in smo na začetku palete – zelena barva
za r = 3 je pozp = 450 in smo na območju svetlo modre barve

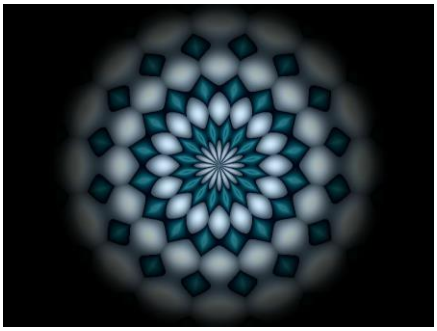
Drugi primer (slika 12):
barva = f(r,fi)
pozp = r * 100 + 30 * (1 + sin(fi * 12)) pozicija za odvzem barve iz palete



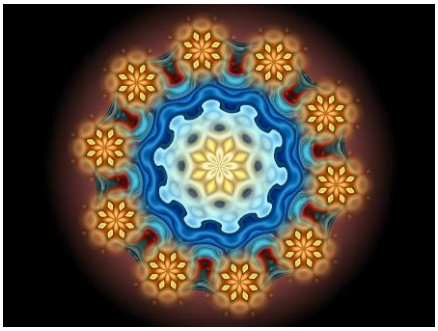
Slika 12

V tem primeru lahko pričakujemo, da bodo koncentrični barvni pasovi deformirani zaradi mnogokratnika kota fi. Lahko tudi predvidimo razporeditev barv od sredine proti robu. Pri uporabi kompleksnejših matematičnih izrazov dobimo izredno zanimive simetrične forme. Z uporabo dodatnih spremenljivk iz procesa pa se simetrija počasi izgublja in dobimo nenavadne abstraktne oblike.

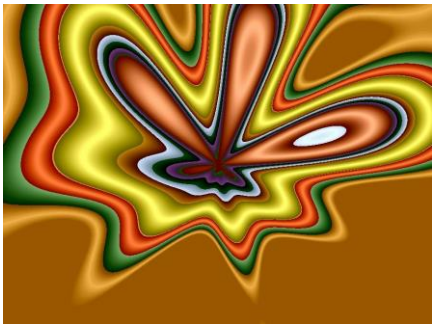
Na slikah 13, 14 in 15 je nekaj primerov slik generiranih s konkretnim programom.



Slika 13



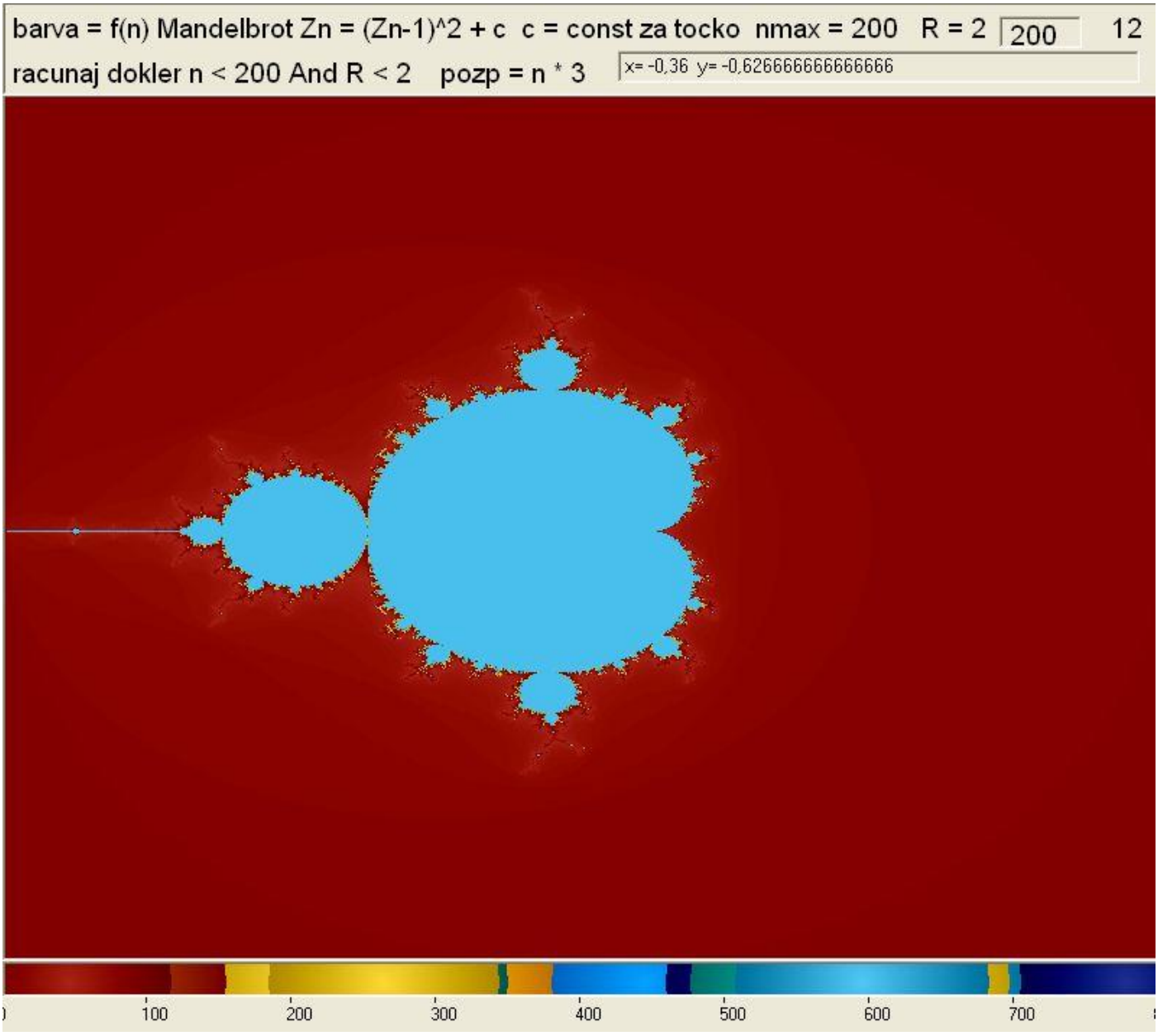
Slika 14



Slika 15

Risanje slike v kompleksni ravnini

Vsaki točki kompleksne ravnine (pixel) lahko priredimo kompleksno število Z = x + yi, kjr je x realna komponenta, y pa imaginarna komponenta. Določimo velikost, in sicer x od -2 do +2, y od -2 do +2. Za kompleksna števila velja zakonitost $i^2 = -1$ in iz tega izhajajo vse zakonitosti računanja. Za vsako točko v ravnini lahko izvajamo iteracijo izraza $Z(n) = Z(n-1)^2 + c$ ter preverjamo, ali vrednost izraza za neko točko c divergira ali konvergira, ob pogoju Z(0) = 0. Točke, za katere zaporedje konvergira, pripadajo posebni množici, ki jo imenujemo Mandelbrotova množica (po matematiku Benoitu Mandelbrotu). Meja med točkami Mandelbrotove množice in točkami, za katere iteracija divergira, je Mandelbrotov fraktal. Najbolj preprosta, neformalna definicija fraktala bi lahko bila naslednja: fraktal je nekaj, kar se ponavlja v neskončnost, nekaj, kar se nikoli ne konča in se spreminja z istim procesom znova in znova. Večkrat se uporablja tudi naslednja definicija: fraktali so geometrijski objekti, katerih fraktalna dimenzija je veliko večja od topološke dimenzije. Fraktalno dimenzijo lahko opazujemo tako, da fraktal povečujemo preko vseh meja, kar nam omogočajo šele računalniki z grafičnimi programi. Pri povečevanju opazimo samopodobnost oblik, kar je tipična značilnost fraktalov. Večkrat se omenja tudi primer merjenja dolžine obale: meritev lahko izvajamo po obalni cesti, lahko gremo okrog skal, dalje okrog vsakega kamna in tako pridemo do najmanjših delcev materije. Vsaka od teh meritev bo seveda daljša od prejšnje in v idealiziranem matematičnem modelu nikdar ne pridemo do konca. Tudi v naravi se pojavljajo preproste fraktalne oblike, na primer praprot, cvetača, oblaki, snežinke in druge strukture s ponavljajočimi se oblikami.

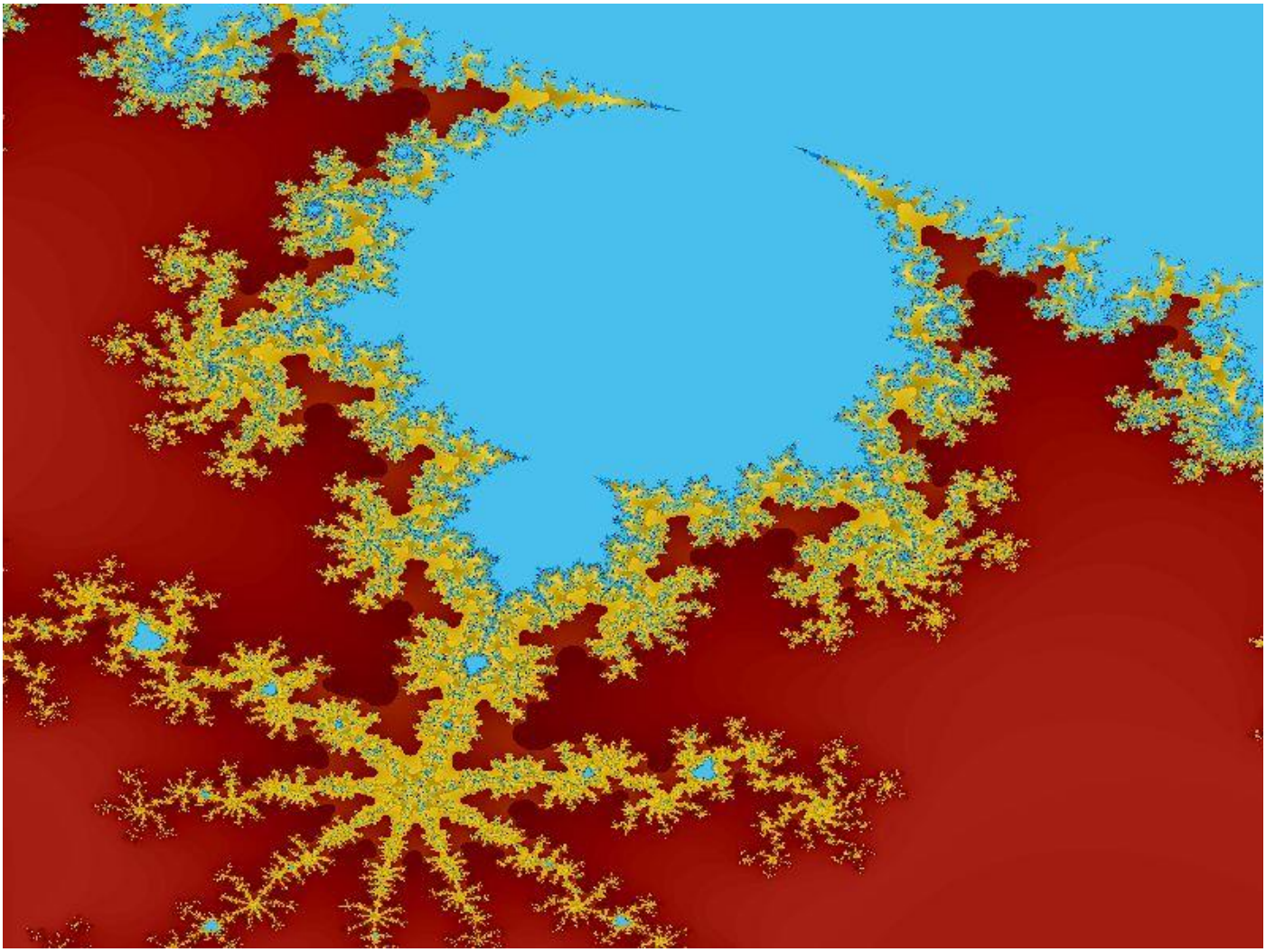


Slika 16

Risanje fraktalov je praktično možno samo z uporabo računalnika in ustreznega programa. Na sliki 16 je primer Mandelbrotovega fraktala, narisanega z enim od mojih programov.

Rišemo v kompleksni ravnini in tudi v tem primeru bom uporabil barvno paleto za določanje barve točke. Na vsaki točki ravnine izvajam iteracijo izraza $Z(n) = Z(n-1)^2 + c$. Iteracija se začne z $Z(0) = 0$, konstanta c pa je kar kompleksno število, ki pripada trenutni točki, katere barve bo program določil. Za preverjanje, ali vrednost izraza divergira ali konvergira si postavim naslednje omejitve: če je po 200 iteracijah absolutna vrednost izraza manj kot 2, potem rečem, da točka konvergira in je c element Mandelbrotove množice. Če pa vrednost izraza preseže 2 (pobegne), to pomeni, da zaporedje divergira in c ni element Mandelbrotove množice. Število iteracij uporabim za določanje barve točke in to tako, da izračunam pozicijo za odvzem barve iz barvne palete (v mojem primeru je to pozp = trikratnik števila iteracij)

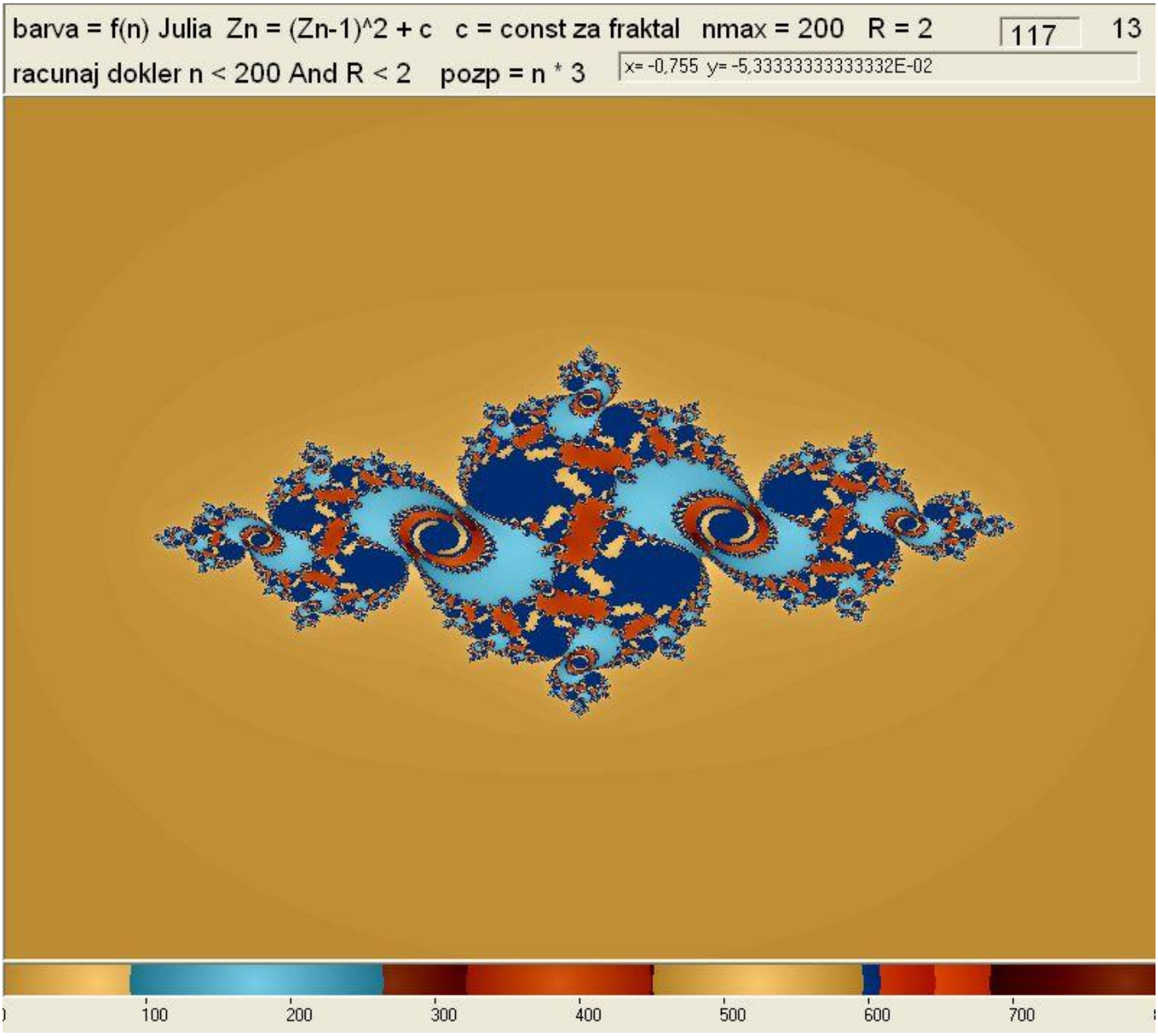
Modro polje so točke, ki pripadajo Mandelbrotovi množici, in sicer za njih velja, da je bilo doseženo število iteracij 200 in po formuli pozp = n * 3 dobim vrednost 600. Na poziciji 600 na barvni paleti dobim svetlo modro barvo. Točke obarvane z različnimi niansami bordo-rjave barve imajo relativno malo iteracij (vrednost izraza je že po nekaj iteracijah presegla 2). Torej jemljem barvo z začetka barvne palete. Zanimive pa so točke na meji fraktala, ki se niti ne vidijo, vendar pa imajo zelo različna števila iteracij. Na sliki 17 je 50-krat povečan spodnji rob fraktala in tu se že vidijo značilne fraktalne oblike. Teoretično gre povečevanje fraktalov preko vseh meja, omejitev postanejo računalniške zmožnosti obdelave numeričnih vrednosti.



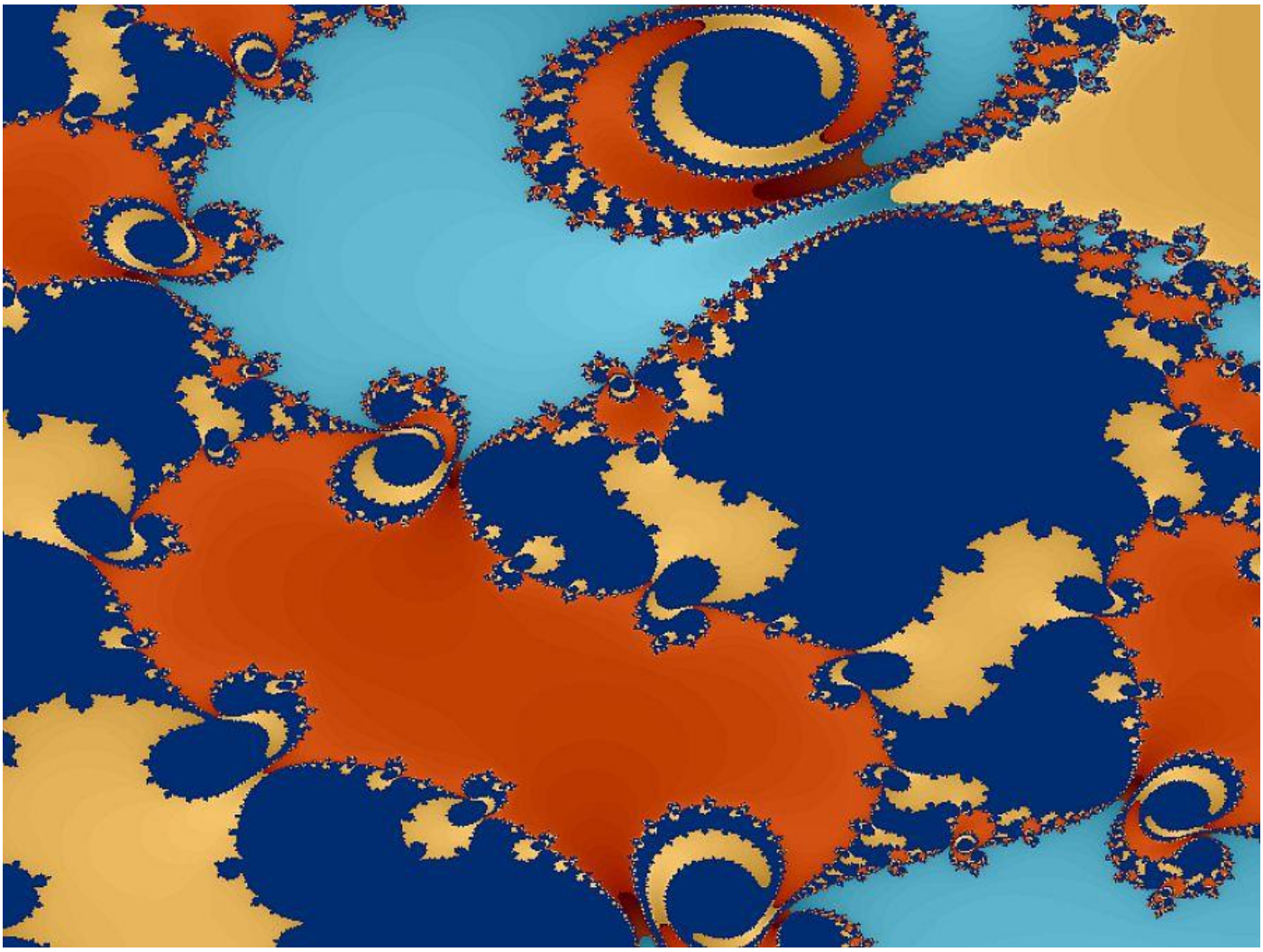
Slika 17

Doslej je bilo govora le o Mandelbrotovem fraktalu, kateremu pa je neposredno soroden Juliajev fraktal, ki je dobil ime po matematiku Gastonu Juliaju. Za določen koordinatni sistem obstaja en sam Mandelbrotov fraktal, ki v grafičnem smislu variira le glede na barvno paleto. Vsaki točki Mandelbrotovega fraktala pa lahko priredimo en Juliajev fraktal, kar teoretično pomeni, da se za Mandelbrotovim fraktalom skriva neskončno Juliajevih podmnožic oziroma Juliajevih fraktalov. Zadevo sem preveril s posebnim programom, ki omogoča risanje obeh fraktalov.

Pri risanju Juliajevega fraktala uporabljamo isti princip iteracije kot pri Mandelbrotu, vendar s to razliko, da se iteracija začne pri kompleksnem številu Z(0), ki odgovarja trenutni točki, konstanta c pa je v naprej določena in je ista za celoten fraktal oziroma za vse iteracije, ki se začnejo v poljubni točki v kompleksni ravnini. Problem Juliajevega fraktala je v vnaprejšnjem izboru konstante c (tej konstanti rečemo tudi parameter Julijevega fraktala). Zanimive grafične oblike dobimo, če izberemo točko na meji med Mandelbrotovo množico in ostalimi točkami. Na sliki 18 je primer Juliajevega fraktala s parametri x = -0.755 in y = -5.33333E-02 in z uporabo pripadajoče barvne palete. Seveda je bil predhodno ustvarjen Mandelbrotov fraktal in izbrana točka na njegovi meji. Na sliki 19 je prikazana 50-kratna povečava izbranega segmenta (skrajno desni del osnovnega fraktala). Temno modra polja so elementi Juliajeve množice (200 iteracij pomeni lokacijo 600 na barvni paleti).



Slika18

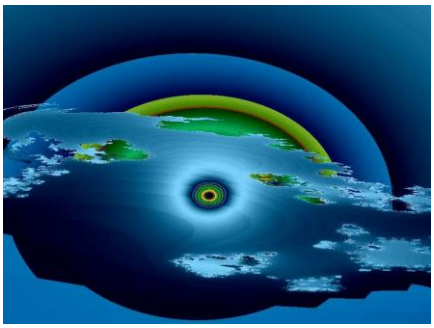


Slika 19

Kakšno povezavo imajo sedaj fraktali z generiranjem slik, saj so že sami po sebi zanimive in lepe grafične podobe? Tu sem v svojih programih naredil zanimiv korak naprej in sicer v tem smislu, da sem deformiral osnovni fraktalni izračun z dodatnimi matematičnimi operacijami. Na ta način sem v veliki meri odpravil samopodobnost pri povečevanju in se znebil ponavljajočih se geometrijskih oblik. Tak program lahko generira slike, ki po svojem videzu ne spominjajo na fraktale, in to je bil tudi moj namen. Na slikah 20, 21 in 22 so trije primeri tovrstno generiranih slik.

Slika 20

Slika 21



Slika 22