Razvoj aplikacije za navođeno generiranje jednostavnih grafika uporabom genetskog algoritma

Cacan, Benjamin

SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET ZAGREB

ZAVRŠNI RAD

Benjamin Cacan

SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET ZAGREB

Smjer: tehničko-tehnološki

ZAVRŠNI RAD

Razvoj aplikacije za navođeno generiranje jednostavnih grafika uporabom genetskog algoritma

Mentor:
doc. dr. sc. Tibor Skala

Student:
Benjamin Cacan

Zagreb, 2019.

SAŽETAK

U teorijskom dijelu ovaj rad opisuje ključne dijelove genetskog algoritma uspoređujući procese algoritma s procesima prirodne selekcije kako bi objasnio princip rada i svrhu upotrebe genetskog algoritma pri rješavanju složenih problema. Primjenom interaktivne selekcije, za razliku od one klasične, prikazuje se mogućnost primjene genetskog algoritma i u grafičkom dizajnu – generativni dizajn.

Praktični dio rada opisuje procese genetskog algoritma funkcijama u JavaScript programskom jeziku uz p5.js biblioteku. Navedeni alati koriste se za razvoj web aplikacije koja korisniku daje mogućnost navođenog generiranja jednostavnih grafika, to jest upravljanja evolucijskim procesom koji generativni dizajn prilagođava korisničkim preferencijama.

Ključne riječi: genetski algoritam, prirodna selekcija, generativni dizajn

ABSTRACT

In the theoretical part, this paper describes the key parts of a genetic algorithm by comparing algorithm processes with natural selection processes to explain the working principle and purpose of using a genetic algorithm to solve complex problems. By using interactive selection, unlike classical selection, the possibility of applying a genetic algorithm in graphic design is shown – generative design.

The practical part of the paper describes the genetic algorithm processes with functions in JavaScript programming language with p5.js library. These tools are used to develop a web application that gives the user the ability to generate simple graphics, that is, the ability to manage an evolutionary process that adjusts the generative design to the user's preferences.

Keywords: genetic algorithm, natural selection, generative design

SADRŽAJ

1. UVOD

Genetski algoritam predstavlja metodu optimizacije pri traženju rješenja za određeni problem - po uzoru na osnovne principe prirodne selekcije, grubo ih oponašajući.

Evolucija u prirodi je proces koji traži najbolje jedinke, one s najvećom sposobnosti prilagodbe na uvjete u prirodi. Najsposobnije jedinke dominiraju u prirodi reproducirajući se i prenose svoje gene budućoj generaciji jedinki, a geni slabijih jedinki izumiru jer nemaju potomstva koje bi ih preuzelo. Pri reprodukciji dolazi do rekombinacije gena što rezultira time da će u novoj generaciji jedinke iste vrste biti različite jedne od drugih te od onih iz prošle generacije, ali će biti i slične svojim roditeljskim jedinkama (iz prošle generacije). Ova izmjena gena se u genetskim algoritmima naziva križanje. Uz križanje, u vrlo maloj mjeri dolazi i do mutacije. Mutacija predstavlja slučajnu promjenu genetskog materijala u genetskom zapisu neke jedinke (djelovanjem vanjskih utjecaja). Proces odabira najsposobnijih jedinki zove se selekcija, a procesi križanja i mutacije se kod genetskih algoritama nazivaju genetski operatori.

Teorijski dio rada opisuje tijek genetskog algoritma te prolazi kroz sve procese potrebne za uspješan evolucijski proces služeći se genetskim algoritmom. Praktični dio zatim prikazuje kako se procesi genetskog algoritma mogu definirati kao funkcije u programskom jeziku JavaScript kako bi se genetski algoritam koristio unutar web aplikacije te kako pomoću genetskog algoritma generirati jednostavne grafike ovisno o korisničkoj interakciji.

Web aplikacija za navođeno generiranje jednostavnih grafika nalazi se na adresi: http://benjamin-cacan.from.hr/ga/

2. TEORIJSKI DIO

2.1. Prednost genetskog algoritma

Genetski algoritam služi kako bi se došlo do rješenja nekog problema za koji postoji tako veliki broj mogućih odgovora da bi klasična metoda pokušaja i pogrešaka trajala predugo da bi u realnom vremenu našli točno rješenje određenog problema.

Naprimjer, ako osoba A zamisli jedan grad na svijetu, osoba B nema realnu šansu doći do rješenja (grad je relativan pojam na području svijeta, samim time nije poznat ukupan broj gradova na svijetu) te može samo nagađati na sreću. No ako osoba A odgovora na odgovore osobe B, naprimjer s toplo/hladno ili s koliko je ponuđen odgovor udaljen od točnog rješenja - tada osoba B dobiva ocijenjene odgovore te se može sa svakim sljedećim odgovorom (evoluirajući) sve više približiti rješenju problema, to jest točnom gradu osobe A.

Najveća prednost genetskog algoritma je to što po svojoj naravi radi paralelno. Dok većina algoritama radi serijski i može razvijati rješenje problema samo na jedan način, ako se to rješenje ne pokaže kao optimalno algoritam mora odbaciti uložen rad i početi tražiti rješenje ispočetka. S obzirom na to da genetski algoritam paralelno (istovremeno) razvija različita rješenja, u slučaju da se neko rješenje ili niz rješenja pokaže kao neisplativo genetski algoritam jednostavno eliminira ta rješenja i nastavlja raditi s onima koja se čine isplativima dajući im tako veću vjerojatnost uspjeha u potrazi za optimalnim rješenjem. [1]

Sljedeći primjer jest teorem beskonačnih majmuna koji glasi: Majmun koji nasumično piše po tipkovnici pisaćeg stroja beskonačno dugo, u konačnici će napisati bilo koji željeni tekst, kao npr. sva djela Shakespearea. Ali, u praksi, šansa da se to dogodi je toliko mala da nema stvarnog vremenskog raspona u kojem bi se to moglo dogoditi. Sljedeći račun će to i dokazati.

Uzme li se samo jedan Shakespeareov citat (pojednostavljen radi primjera): „*biti ili ne biti*” (ukupno 16 znakova), i odredi se da majmun iz teorema piše na pisaćem stroju s hrvatskom abecedom i razmakom (ukupno 31 znak). Vjerojatnost da jedno točno slovo (znak) na pisaćem stroju bude pritisnuto iznosi 1/31. Sukladno tome, vjerojatnom da

izabrani citat bude točno utipkan iznosi $(1/31)^{16}$ što je jednako $\frac{1}{72742312174718526382848 1}$. Što znači da kad bi majmun iz primjera svake sekunde utipkao jednu rečenicu od 16 znakova - trebalo bi mu nepraktično mnogo godina da sa sigurnošću točno utipka citat od samo 4 kratke riječi – „*biti ili ne biti*".

Ovim prikazanim tempom, u slučaju kompleksnih problema, pomoću prezentirane metode brzo bi prešli brojku od 13.8 milijardi godina (brojka koja označava i starost svemira) potrebnog vremena kako bi se došlo do traženog rješenja. Može se zaključiti da se metodom pogađanja svakog mogućeg rješenja nikada (u realnom vremenu) neće doći do točnog rješenja određenog problema već da pokušaji rješenja moraju na neki način evoluirati.

Genetski algoritam, stoga, vrlo dobro rješava probleme koji imaju zbilja velik prostor u kojem se nalazi rješenje, prevelik za opsežno pretraživanje u bilo kojem realnom vremenu. Većina takvih problema su nelinearni problemi. Kod linearnih problema sposobnost svakog elementa je neovisna o ostalima stoga svako poboljšanje bilo kojeg dijela rezultira poboljšanjem sustava u cjelini. No samo nekolicina stvarnih problema je linearno. Nelinearnost je uobičajena, promjena jednog elementa može uvelike utjecati na cijeli sustav, a višestruke promjene koje su na individualnoj razini štetne po element mogu dovesti do puno uspješnijeg napretka sposobnosti algoritma. [1]

Nelinearnost rezultira velikom kombinatorikom: (primjer) Ako je problem linearan, 1000 znamenkasti binarni niz može se opsežno pretražiti procjenjujući samo 2000 kombinacija, no ako se radi o nelinearnom problemu, opsežna pretraga tada zahtijeva procjenu 2^{1000} kombinacija (brojka ima 302 znamenke). [1]

2.2. Princip rada genetskog algoritma

Genetski algoritam funkcionira po uzoru na evoluciju u prirodi (Darwinova teorija evolucije), no ne iz razloga kako bi se precizno prepisao cijeli model prirodne evolucije u svrhu proučavanje bioloških procesa. Genetski algoritam samo koristi ideje prirodne evolucije i genetike primjenjujući ih u izradi softvera za rješavanje određenog problema. Osnovni principi procesa evolucije isti su u prirodi i genetskom algoritmu, a to su: nasljednost, varijacija, selekcija.

Nasljednost osigurava da djeca nasljeđuju svojstva roditelja. Određene jedinke (rješenja) u novoj generaciji imaju dio karakteristika istih kao određene jedinke (rješenja) iz prošle generacije. Jedinke koje nisu izabrane da sudjeluju u reprodukciji neće prenijeti svoje gene (podatke) jedinkama u sljedećoj (novoj) generaciji.

Varijacija zahtjeva raznovrsnost osobina (svojstva) jedinki kako bi se osiguralo da populacija jedinki evoluira kroz generacije. U populaciji gdje nema varijacije, sve jedinke imaju identična svojstva te nikad neće doći do nastanka jedinki s novim kombinacijama svojstava, već će sve nove jedinke (djeca) biti potpuno jednake starim jedinkama (roditelji).

Kada bi sve jedinke u generaciji imali jednake šanse reproducirati se evolucija ne bi bila uspješna te kvaliteta populacije ne bi ni na koji način rasla. Selekcija brine da bolje prilagođene jedinke iz populacije, one ocijenjene kao vrjednije, imaju veću vjerojatnost reproducirati se i prenijeti svoje gene, za razliku od slabijih jedinki koje i dalje imaju šansu reproducirati se, ali je ona puno manja. Ovaj mehanizam se može nazvati i „opstanak najjačih”, iako to nije u svakom slučaju doslovno točno jer u prirodi preživljavaju oni najbolje prilagođeni okolini, a to ne moraju biti fizički najjače, najbrže ili najveće jedinke. [2]

2.3. Proces rada genetskog algoritma

Dijagram toka 1 – Proces genetskog algoritma

Proces selekcije (biranje dva roditelja) i reprodukcije (križanje i mutacija) ponavlja se N puta sve dok se ne dobije nova populacija od N novih elemenata. Zatim ta nova populacija sačinjena od novih elemenata (djece) postaje trenutna populacija te se ponovo računa *fitness* trenutne populacije te se provodi selekcija i reprodukcija. Navedeni procesi se ponavljaju u petlji sve dok ne bude ispunjen određeni uvjet za kraj evolucijskog procesa ili dok korisnik ne odluči prekinuti evolucijski proces.

2.4. Fitness funkcija

Fitness funkcija naziva se još i funkcija dobrote ili funkcija ocjene kvalitete jedinke, funkcija sposobnosti, funkcija cilja. [3] To je ključna funkcija u procesu selekcije dobrih elemenata iz populacije. *Fitness funkcija* je u većini slučajeva jednadžba koja daje vrijednost svakom elementu iz populacije, ocjenjuje koliko je koji element kvalitetan, to jest koliko je blizu željenom cilju razvijanog algoritma. Uloga ove funkcije je da osigurava kvalitetnu selekciju elemenata za reprodukciju. Zato je *fitness* vrijednost izračunata za svaki element proporcionalna vjerojatnosti tog elementa da bude izabran za proces reprodukcije.

Ova funkcija iznimno je bitan dio genetskog algoritma. Kako bi se izradila uspješna *fitness funkcija* moraju se odrediti ciljevi određenog problema koji se pokušava riješiti algoritmom. Zatim je potrebno odrediti metodu vrednovanja kako bi se u konačnici, jednadžbom, numerički ocjenjivalo elemente iz populacije (moguća rješenja problema).

Sljedeće će poglavlje na jednostavnom primjeru prikazati kako se *fitness funkcija* može odrediti i poboljšati kako bi evolucijskim procesom u konačnici dobili što bolje rezultate.

2.4.1. Primjer određivanja *fitness funkcije*

Određivanje fitness funkcije za algoritam koji generira jedan naslov za plakat:

Naslov za plakat generira se u jednoj nasumičnoj boji (u HSB sustavu boja) s nasumičnom veličinom pisma. Naslov za plakat treba biti u boji što moguće veće saturacije i svjetline te što veće veličine pisma, ali da se naslov ne prelama u novi redak.

(U svrhu primjera: Naslov za plakat se prelama ako je pismovna veličina veća od 85 tipografskih točaka.)

Primjer jednadžbe za *fitness funkciju* (linearna)

$$f(naslov) = \frac{S(boja) + B(boja) + size(naslov)}{1 + prijelom(naslov)}$$

S(boja) - saturacija boje [0-100]
B(boja) - svjetlina boje [0-100]
size(naslov) - pismovna veličina naslova
prijelom(naslov) - prelazili li naslov u novi red: ne prelazi ili prelazi [0 ili 1]

Tablica 1 – *Fitness* vrijednosti dobiveni linearnom *fitness funkcijom*

S(boja)	B(boja)	size(naslov)	prijelom(naslov)		**Fitness vrijednost**
59	67	66	0		**192**
85	72	152	1		**154**
15	44	75	0		**134**
93	52	55	0		**200**

Kako rezultati *fitness funkcije* direktno utječu na vjerojatnost reprodukcije elemenata ocijenjenih ovom funkcijom - bitno je *fitness* funkciju što preciznije odrediti. Jedan od načina za poboljšanje funkcije jest to da se koristi eksponencijalna funkcija umjesto linearne (iz gornjeg primjera). [2]

Primjer jednadžbe za *fitness funkciju* (eksponencijalna)

$$f(naslov) = \left[\frac{[S(boja) + B(boja) + size(naslov)]^2}{[1 + prijelom(naslov)]^2}\right] / 200$$

Tablica 2 – *Fitness* vrijednosti dobiveni eksponencijalnom *fitness funkcijom*

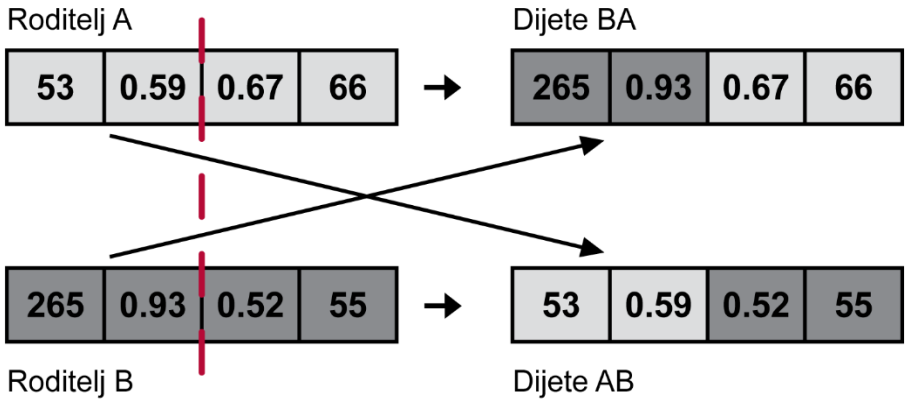
S(boja)	B(boja)	size(naslov)	prijelom(naslov)		**Fitness vrijednost**
59	67	66	0		**184**
85	72	152	1		**119**
15	44	75	0		**90**
93	52	55	0		**200**

U ovom primjeru vidi se kako i najjednostavnija eksponencijalna jednadžba čini veće razlike između bliskih rezultata naspram linearne jednadžbe. U oba primjera dane su iste vrijednosti za izračun *fitnessa* generiranih naslova za plakat. Razlika *fitness* vrijednosti između dva najveća rezultata (200 i 192) u prvom primjeru je 4.08%, a razlika između dva najveća rezultata (200 i 184) u drugom primjeru je 8.33%. Također razlika između dva najmanja rezultata u prvom i drugom primjeru porasla je s 13.89% na 27.75%. S obzirom na to da se za jednake ulazne veličine (prvi i drugi primjer) dobivaju veće razlike između rezultata koristeći eksponencijalnu *fitness funkciju*; To pogoduje procesu selekcije jer će elementi s vrlo sličnom uspješnošću biti drastičnije ocjenjivani dajući elementu koji je malo bolji od drugog uvećanu prednost. Na taj način izbjegava se to da elementu koji ima malo manju uspješnost od nekog drugog i tom drugom elementu u konačnici budu dodijeljena jednaka vjerojatnost za reprodukciju.

2.5. Križanje

Križanje kao podproces reprodukcije definira kako novonastali element preuzima (nasljeđuje) genetski materijal (gene) od elemenata iz prethodne populacije. U procesu križanja sudjeluju dva elementa (jedinke) koji se nazivaju „roditelji“, križanjem se stvaraju jedan ili dva nova elementa koji se nazivaju „djeca“. Novonastali element posjeduje gene oba „roditelja“, kombinaciju njihovih osobina. Križanjem dva elementa dovoljno sposobna elementa (većih fitness vrijednosti) očekuje se da dobiveno „dijete“ bude element barem jednako sposoban kao roditelji, no moguće je i da loše osobine jednog roditelja budu zamijenjene dobrim osobinama drugog pa tako njihovo „dijete“ bude još sposobnije od njih samih. [3]

Postoji više načina kako se mogu križati geni, ovisno o problemu različite metode križanja dati će bolje ili lošije rezultate. Jedan od načina jest da se nasumično odabere točka prekida genetskog zapisa koji se križa i ostatak tog genetskog zapisa zamjeni se s ostatkom genetskog zapisa drugog roditelja.



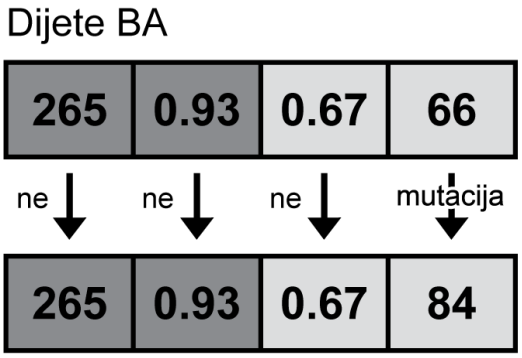
Slika 1 – Križanje dva elementa (roditelja)

(vrijednosti iz poglavlja Primjer određivanja *fitness funkcije*)

2.6. Mutacija

Nakon križanjem dobivenog genetskog materijala za novi element („dijete“), preostalo je provesti proces mutacije prije dodavanja novog element u novu populaciju. Mutacija može biti proizvoljan korak u procesu reprodukcije elemenata. Mutacija se postoji zbog načela varijacije; S obzirom da je početna populacija generirana potpuno nasumično pretpostavlja se da će se elementi iz te populacije međusobno razlikovati, no koliko varijacije ima u početnoj populaciji je ujedno i najviše što se može postići. Zato se uvodi proces mutacije koji omogućava dodatne varijacije tijekom evolucijskog procesa.

Mutacija je slučajna promjena gena, a opisuje se kao vjerojatnost da se promijeni gen (npr. 5%, 1%, 0,1%). Uzme li se primjer iz prošlog poglavlja i zada vjerojatnost mutacije od 5% znači da će svaki gen iz genetskog zapisa dobivenog križanjem imati 5% vjerojatnosti da mutira. U ovom slučaju mutiranje gena znači da će umjesto trenutne vrijednosti gen dobiti novu nasumičnu vrijednost. Kako se u primjeru radi o genetskom zapisu od 4 gena, u 80% slučajeva neće doći do promjene niti jednog gena, no kada ipak dođe do toga gen mutira. (Slika 2)



Slika 2 – Mutacija genetskog zapisa

Vjerojatnost mutacije može uvelike utjecati na cijeli sustav evolucije. Visoka vjerojatnost mutacije (npr. 80%) bi negirala evolucijski proces, ako se većina gena nasumično generira nije sigurno da će se uspješniji geni sve više pojavljivati svakom uzastopnom generacijom. [2]

2.7. Interaktivna selekcija

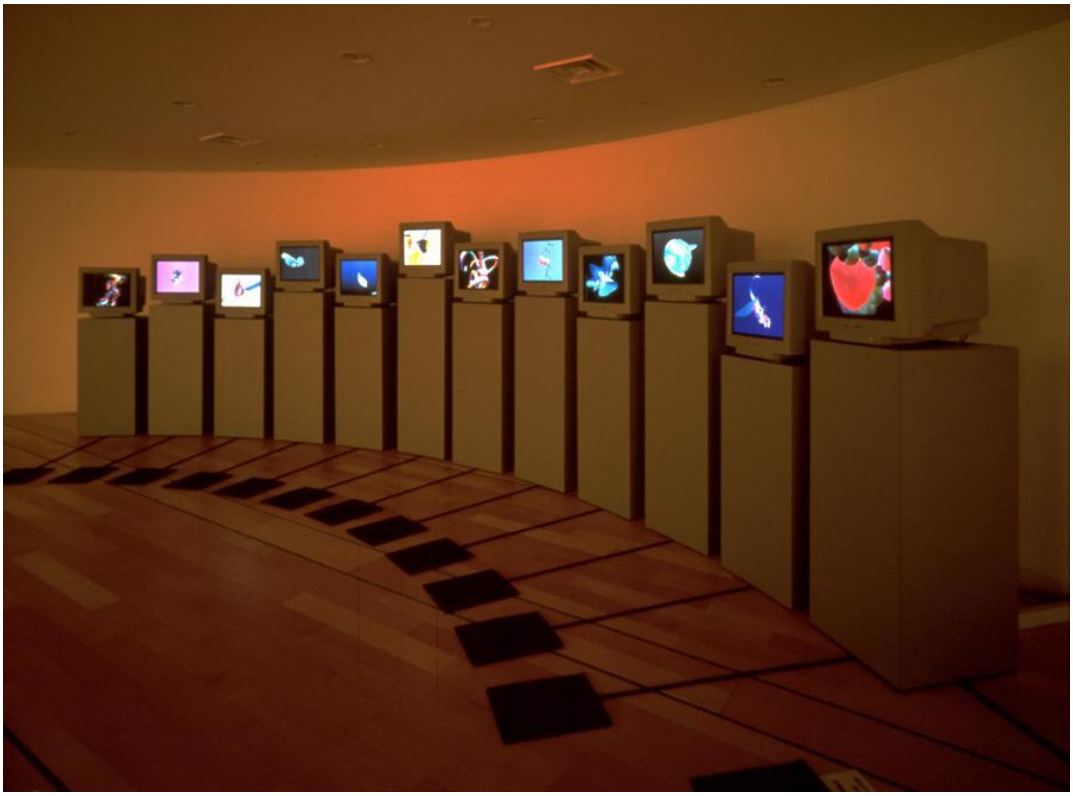
Osim za rješavanje strogo računalnih problema, genetski algoritam ima široko područje primjene. Jedna od primjena algoritma je i u vizualnoj umjetnosti upotrebom interaktivne selekcije. Interaktivna selekcija predstavlja proces evoluiranja nekog rješenja (često računalno generirane slike) kroz korisničku interakciju. Savršeni primjer bio bi muzej s deset slika, posjetitelj bi odabrao koje su slike njemu najbolje, a algoritamski proces bi na osnovu toga generirao (evoluirao) nove slike na temelju posjetiteljevih preferencija. [2]

Razlika između klasičnog genetskog algoritma i onog s interaktivnom selekcijom jest taj da interaktivni nema fitness funkciju u obliku matematičke formule, umjesto toga traži se od korisnika da dodijeli vrijednosti generiranim elementima. Ovaj pristup s jedne strane može biti povoljan jer je dobru fitness funkciju općenito nije lako definirati, a za probleme gdje je rješenje (na primjer) vizualno - definiranje formule za fitness funkciju postaje iznimno teško. Mana ove metode je ta što se u proces dodaje ljudski faktor čime se cijeli proces značajno usporava. Populacija mora imati dovoljno malo članova da ih čovjek može vizualno pregledati i ocijeniti, za to vrijeme cijeli proces stoji i čeka na čovjeka za razliku od klasične selekcije gdje računalo u malom vremenskom roku odradi veliki broj instrukcija obrađujući podatke.

2.7.1. Interaktivna muzejska instalacija „*Galápagos*“

Karl Sims, umjetnik i istraživač računalne grafike, autor je interaktivne instalacije *Galápagos* koja se služila interaktivnom selekcijom, to jest genetskim algoritmom gdje su fitness vrijednost dodjeljivali posjetitelji galerije. Instalacija *Galápagos* izložena je u Tokiju (Japan) 1997. - 2000. godine te u Lincolnu (UK) 1999. godine. Instalacija je na interaktivni način prikazivala Darwinovu evoluciju virtualnih organizama, u prostoriji se nalazilo 12 računala (monitora) na kojima su se prikazivali animirani apstraktni oblici organizama. Posjetitelji su sudjelovali u izložbi na način da odaberu koji organizmi su im najzanimljiviji i stanu na senzor odabranog organizma. Ispred svakog monitora se nalazio jedan senzor koji je bilježio koliko dugo posjetitelj stoji na njemu. Odabrani organizmi su preživljavali reproducirali se i mutirali, a oni neodabrani bi zamijenjeni s novonastalim organizmima. Kako su reproducirani organizmi mutirali

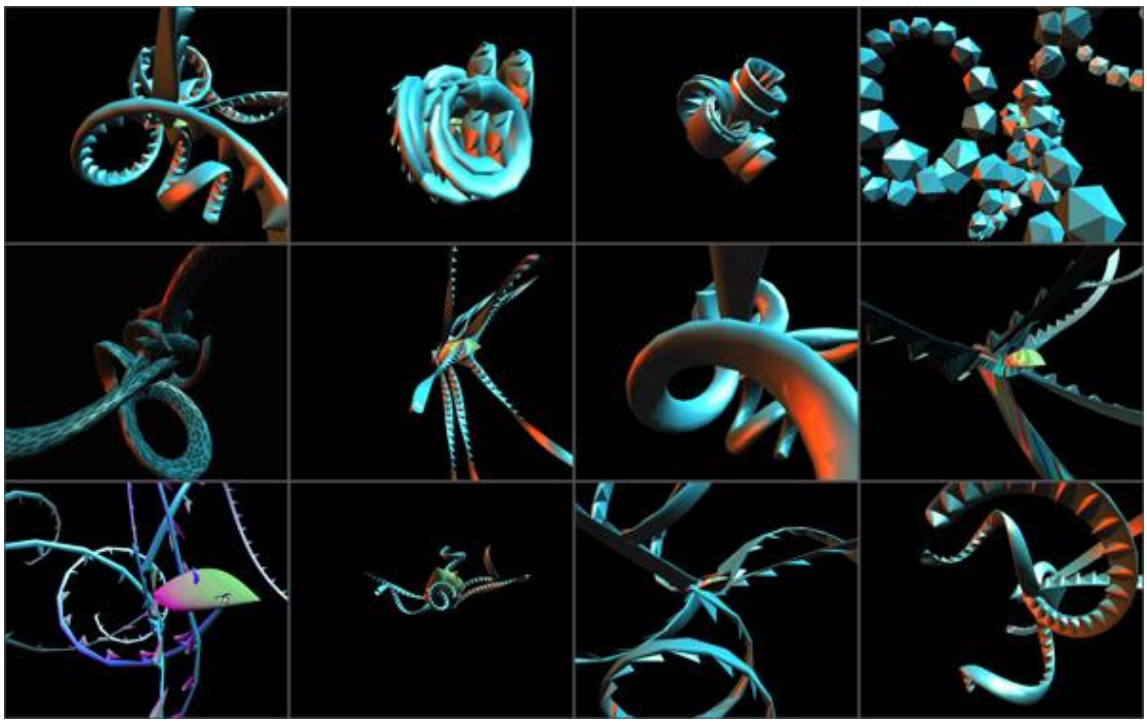
nasumičnom vjerojatnošću, ponekad je mutacija rezultirala pogodnijim organizmima koji su bili zanimljiviji od svojih prethodnika, a kako se taj evolucijski ciklus reprodukcije i selekcije nastavljao - pojavljivalo se sve više zanimljivih organizama. [4]



Slika 3 – Instalacija *Galápagos*, Karl Sims (Fotografija: OHTAKA Takashi)
(izvor: https://www.ntticc.or.jp/en/archive/works/galapagos/)

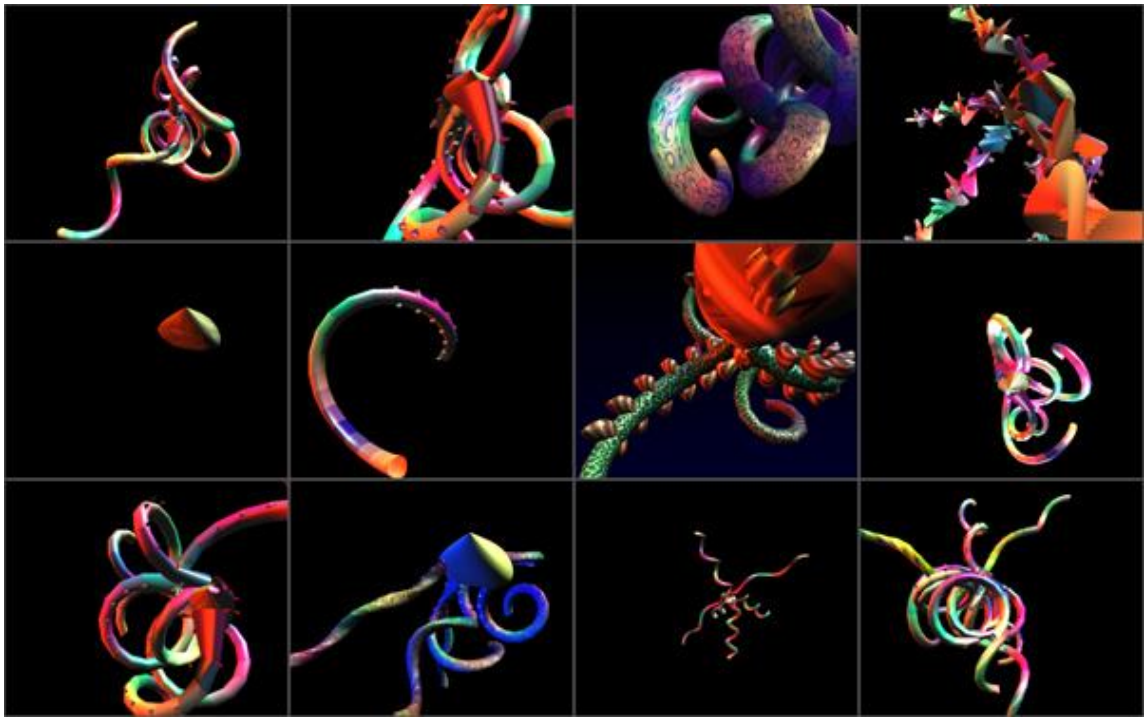
Karl Sims navodi kako je ovaj način interaktivne evolucije koristan jer može biti alat koji daje rezultate kakve ne bi bilo moguće proizvesti na niti jedan drugi način i da daje jedinstvenu metodu za proučavanje evolucijskih sustava.

„*Proces ove izložbe suradnja je između čovjeka i računala. Posjetitelji pružaju estetske informacije birajući koji animirani oblici su najzanimljiviji, a računalo pruža mogućnost simuliranja genetike, razvoja i ponašanja virtualnih organizama. Rezultati potencijalno mogu nadmašiti što čovjek ili računalo može stvoriti samostalno. Iako ukus sudionika određuje rezultate, oni nisi dizajnirani u tradicionalnom smislu. Oni radije koriste selektivni uzgoj za istraživanje 'hiperprostora' mogućih organizama u ovom simuliranom genetskog sustavu. Budući da genetskim kodom i kompleksnosti rezultata upravlja računalo, rezultati nisu ograničeni granicama ljudske sposobnosti dizajniranja ili razumijevanja.*“ [4]



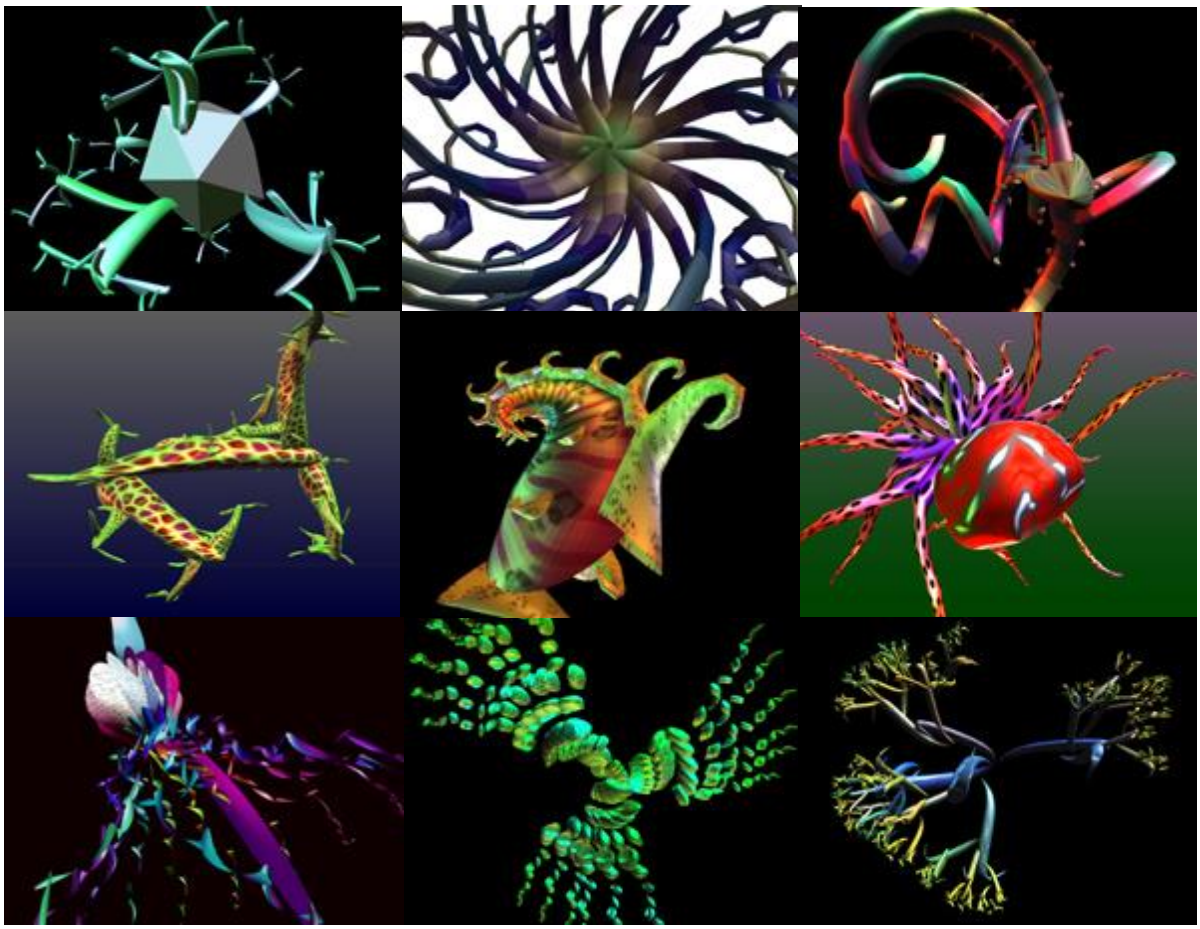
Slika 4 – „Rodbina“ jednog evolucijskog napretka (*Galápagos*, Karl Sims)

(izvor: http://www.karlsims.com/galapagos/galapagos-images.html)



Slika 5 – „Rodbina“ jednog evolucijskog napretka (*Galápagos*, Karl Sims)

(izvor: http://www.karlsims.com/galapagos/galapagos-images.html)



Slika 6 – Organizmi (*Galápagos*, Karl Sims)

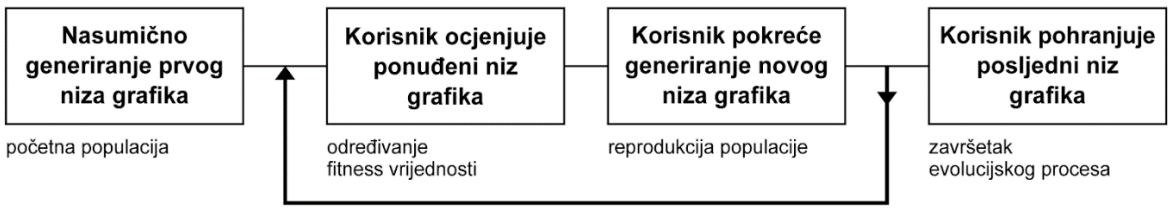
(izvor: http://www.karlsims.com/galapagos/galapagos-images.html)

2. PRAKTIČNI DIO

2.1. Opis aplikacije

Aplikacija za navođeno generiranje jednostavnih grafika upotrebom genetskog algoritma na jednostavan način prikazuje kako se pomoću genetskog algoritma mogu evoluirati i prikazivati generirana grafička rješenja. Aplikacija radi koristeći interaktivnu selekciju genetskog algoritma po uzoru na interaktivnu muzejsku instalaciju „Galápagos“ (Karl Sims). Svrha razvoja ove aplikacije jest izrada alata koji grafičkim dizajnerima može služiti kao model ili primjer alata pomoću kojeg mogu na generirati svoje grafike kako bi postigli određeni vizual ili niz od više vizuala koje ne bi mogli postići nekom drugom tehnikom.

Ova aplikacija generira jednostavne grafike na način da su unaprijed određene instrukcije za iscrtavanje grafike. Aplikacija se sastoji od više različitih kategorija jednostavnih grafika od kojih korisnik bira kategoriju čije grafike želi evoluirati. Svaka kategorija je jedan niz unaprijed određenih instrukcija za iscrtavanje generičkih grafika, no parametri prema kojima se grafike iscrtavaju su neodređeni. Parametri predstavljaju svojstva grafike kao što su: debljina crte, boja, pozicija, veličina grafike, broj točaka za iscrtavanje, broj ponovnih iscrtavanja, određena metoda za iscrtavanje, šansa da se određena metoda promijeni itd. Parametre (unutar zadanih granica) prema kojima se grafike generiraju određuje genetski algoritam, to jest korisnik interaktivnom selekcijom.



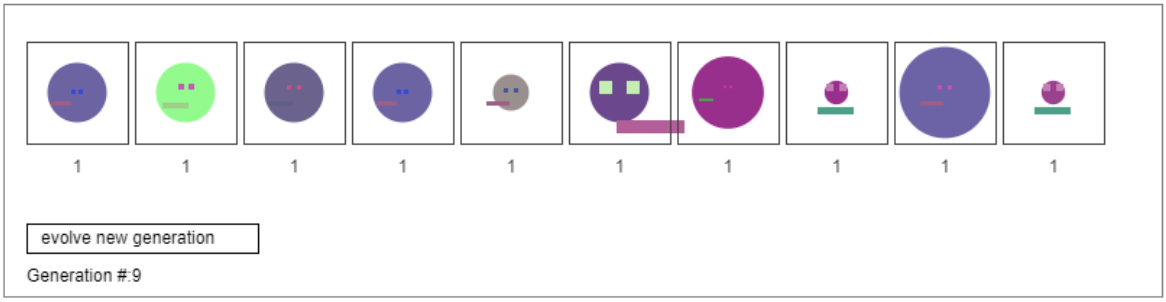
Dijagram toka 2 – Tijek rada aplikacije

2.2. Alati za razvoj aplikacije

Aplikacija je razvijena koristeći JavaScript programski jezik. Prednost JavaScripta je njegova široka dostupnost i sveprisutna podrška. Svaki web preglednik (uključujući i one mobilne!) ima ugrađen JavaScript interpreter, a naredbe se izvršavaju na korisničkoj strani to jest unutar web preglednika. [5]

Uz JavaScript kao temelj za programiranje unutar web stranice, aplikacija je programirana koristeći i p5.js programsku biblioteku (eng. *library*) za JavaScript. P5.js je JavaScript inačica programskog jezika Processing, nudi definirane funkcije za jednostavnije crtanje grafika unutar web stranice, ali nudi i puno više za programiranje web aplikacija, cilj mu je olakšavanje interaktivnog i kreativnog programiranja. Stoga ovaj alat nije namijenjen samo za razvojne programere, već i za umjetnike, dizajnere, istraživače itd. [6]

Programski kod ove aplikacije temeljen je na primjeru interaktivne selekcije Daniela Shiffmana iz knjige *The Nature of Code* koja opisuje simuliranje prirodnih pojava pomoću programskog jezika Processing. [7] Shiffmanov primjer, također prema uzoru na muzejsku instalaciju *Galápagos*, služi samo kao demonstracija ideje o interaktivnoj selekciji. Iako njegov primjer generira trivijalne grafike, svaki element u populaciji je grafika lica sastavljenog od kruga, dva pravokutnika i kvadrata (glava, oči i usta), D. Shiffman uspješno definira osnovne funkcije genetskog algoritma. [2]



Slika 7 – Primjer aplikacije s interaktivnom selekcijom (Daniel Shiffman)

(izvor: http://natureofcode.com/book/chapter-9-the-evolution-of-code/)

2.3. Programiranje genetskog algoritma

Ovo poglavlje prikazat će kako je D. Shiffman definirao genetski algoritam kroz funkcije koje predstavljaju evolucijske procese. [2] Dijelovi genetskog algoritma su univerzalno definirani na način da se iste funkcije mogu koristiti pri razvijanju genetskih algoritama za rješavanje različitih problema.

Početni genetski zapis elemenata (*DNA*) definiran je kao klasa koja sadrži niz od proizvoljnog broja gena gdje je svaki gen nasumične vrijednosti u rasponu 0-1.

```
class DNA {

  constructor(newgenes) {
    let len = 16; // Proizvoljni broj gena u genetskom zapisu
    if (newgenes) {
      this.genes = newgenes;
    } else {
      this.genes = new Array(len);
      for (let i = 0; i < this.genes.length; i++) {
        this.genes[i] = random(0, 1);
      }
    }
  }
...
```

Isječak koda 1 – Definiranje genetskog zapisa

Svaka populacija (*this.population*) je definirana kao klasa koja sadrži: broj elemenata (u jednoj populaciji), vjerojatnost mutacije, niz svih elemenata u trenutnoj generaciji, niz (odabranih gena) za „razmnožavanje“ te brojač generacija.

```
class Population {
  constructor(m, num) {  // num = (broj elemenata)
    this.mutationRate = m;      // m = (vjerojatnost mutacije)
    this.population = [];       // Trenutna populacija
    this.matingPool = [];// Niz za razmnožavanje
    this.generations = 0;       // Redni broj trenutne generacije

     // U populaciju se dodaje određeni broj (num) elemenata (Visual)
        generiranih s genetskim zapisom (DNA)
    for (let i = 0; i < num; i++) {
      this.population[i] = new Visual(new DNA(), 50 + i * 75, 60);
    }
  }
...
```

Isječak koda 2 – Definiranje populacije

2.3.1. Funkcija selekcije

Funkcija sprema *fitness* vrijednosti svakog elementa u populaciji te se dobivene *fitness* vrijednosti skaliraju u raspon 0-1. (11. redak)

Zatim se dobivene vrijednosti normaliziraju u raspon 0-100 i zaokružuju. (13. redak)

S obzirom na fitness vrijednost svaki element iz populacije će biti dodan u niz za „razmnožavanje“ (*matingPool*), s tim da će elementi većeg fitnessa biti dodani više puta, a elementi manje fitnessa manje puta čime se utječe na vjerojatnost da određeni element bude izabran za roditelja, to jest za reprodukciju. (15.-17. redak)

```
1  // Selekcija - generiranje niza za razmnožavanje (matingPool)
2  selection() {
3
4    // Brisanje trenutnog niza za razmnožavanje (matingPool)
5    this.matingPool = [];
6
7    // Računanje ukupnog fitnessa za cijele populacije
8    let maxFitness = this.getMaxFitness();
9
10   for (let i = 0; i < this.population.length; i++) {
11    let fitnessNormal = map(this.population[i].getFitness(), 0, maxFitness, 0, 1);
12
13    let n = floor(fitnessNormal * 100); // Normaliziranje
14
15    for (let j = 0; j < n; j++) {
16      this.matingPool.push(this.population[i]);
17    }
18   }
19 }
```

Isječak koda 3 – Definiranje funkcije selekcije

2.3.2. Funkcija reprodukcije

Funkcija iz niza za razmnožavanje (*matingPool*) nasumično bira dva „roditelja“. (6. i 7. redak)

Genetski materijali izabrana dva roditelja križaju se funkcijom koja je zaduženja za križanje gena (*crossover* funkcija). (18. redak)

Genetski materijal dobiven nakon križanja predstavlja novi element („dijete“) koji zatim prolazi kroz funkciju za mutiranje gena (*mutate* funkcija) kako bi geni novog elementa eventualno mutirali ovisno o vjerojatnosti mutacije (21. redak)

Konačni genetski materijal, to jest novonastali element („dijete“) dodaje se u novu populaciju. (24. redak)

Proces reprodukcije ponavlja se sve dok se nova populacija ne napuni s određenim brojem novih elemenata (veličina populacije je konstantna), zatim se stara populacija zamjenjuje s novom i evolucijski proces se nastavlja s novo dobivenom generacijom.

```
1  // Reprodukcija - stvaranje sljedeće generacije
2  reproduction() {
3    for (let i = 0; i < this.population.length; i++) {
4
5      // Odabir dva roditelja
6      let m = floor(random(this.matingPool.length));
7      let d = floor(random(this.matingPool.length));
8
9      // Pohranjivanje dva izabrana roditelja
10     let mom = this.matingPool[m];
11     let dad = this.matingPool[d];
12
13     // Preuzimanje genetskog zapisa izabranih roditelja
14     let momgenes = mom.getDNA();
15     let dadgenes = dad.getDNA();
16
17     // Križanje preuzetih gena
18     let child = momgenes.crossover(dadgenes);
19
20     // Mutiranje gena novog elementa (djeteta)
21     child.mutate(this.mutationRate);
22
23     // Punjenje nove populacije s novim elementima (djecom)
24     this.population[i] = new Visual(child, 170 + i * 300, 200);
25   }
26   // Brojač generacija se pomiče za +1
27   this.generations++;
28 }
```

Isječak koda 4 – Definiranje funkcije reprodukcije

2.3.3. Funkcija križanja

Funkcija križanja kreira genetski zapis novog elementa („djeteta“) križajući genetske zapise njegovih „roditeljskih“ elemenata.

Ovaj proces križanja nasumično odabire točku prekida genetskog materijala „roditelja“. (7. redak)

U svrhu dobivanja gena za genetski materijal novog elementa („djeteta“) – svaki gen prije točke prekida pripada prvom „roditelju“, a svaki gen nakon točke prekida pripada drugom „roditelju“. (10.-13. redak)

Genetski zapis dobiven križanjem dodjeljuje se novom elementu („djetetu“) i njegova vrijednost se vraća funkciji za reprodukciju kako bi se završio proces reprodukcije tog elementa. (15. i 16. redak)

```
1  // Križanje - kreiranje novog genetskog zapisa
2  crossover(partner) {
3    // Kreiranje novog elementa (djeteta)
4    let child = new Array(this.genes.length);
5
6    // Određivanje točke prekida genetskog zapisa
7    let crossover = floor(random(this.genes.length));
8
9    // Križanje gena roditelja za novi element (dijete)
10   for (let i = 0; i < this.genes.length; i++) {
11     if (i > crossover) child[i] = this.genes[i];
12     else child[i] = partner.genes[i];
13   }
14 // Dobiven genetski zapis vraća se funkciji za reprodukciju
15   let newgenes = new DNA(child);
16   return newgenes;
17 }
```

Isječak koda 5 – Definiranje funkcije križanja

2.3.4. Funkcija mutacije

Funkcija mutacije poziva se unutar funkcije za reprodukciju neposredno nakon funkcije križanja.

Genetski zapis novog elementa dobiven funkcijom križanja prolazi kroz ovu jednostavnu funkciju gdje svaki gen ima mogućnost biti mutiran ovisno o vjerojatnosti mutacije (*m*). Vjerojatnost mutacije jedna je od vrijednosti koja se definira na samom početku genetskog algoritma i ona je konstantna.

U slučaju da neki gen mutira, vrijednost tog gena (raspon 0-1) biti će zamijenjena s nasumičnom vrijednošću u rasponu 0-1.

```
1  // Funkcija mutacije
2  mutate(m) {
3    for (let i = 0; i < this.genes.length; i++) {
4      if (random(1) < m) {
5        this.genes[i] = random(0, 1);
6      }
7    }
8  }
```

Isječak koda 6 – Definiranje funkcije selekcije

2.4. Generiranje jednostavnih grafika

P5.js, dodatna biblioteka za JavaScript, sastavljena je od unaprijed definiranih funkcija koje služe kako bi se na brži i jednostavniji način crtalo unutar HTML *canvasa*, simulirao animacije, interaktivne programe itd. Nazivi funkcija su vrlo intuitivni, na primjer: *line* (iscrtavanje linije), *stroke* (boja linije), *strokeWeight* (debljina linije), *fill* (boja ispune), *rect* (iscrtavanje kvadrata). [8] U svrhu ove aplikacije koristi se i SVG *renderer* koji omogućava opciju spremanja iscrtanog *canvasa* u SVG format što korisniku daje mogućnost da generirane vizuale koristi za daljnji rad unutar grafičkih programa.

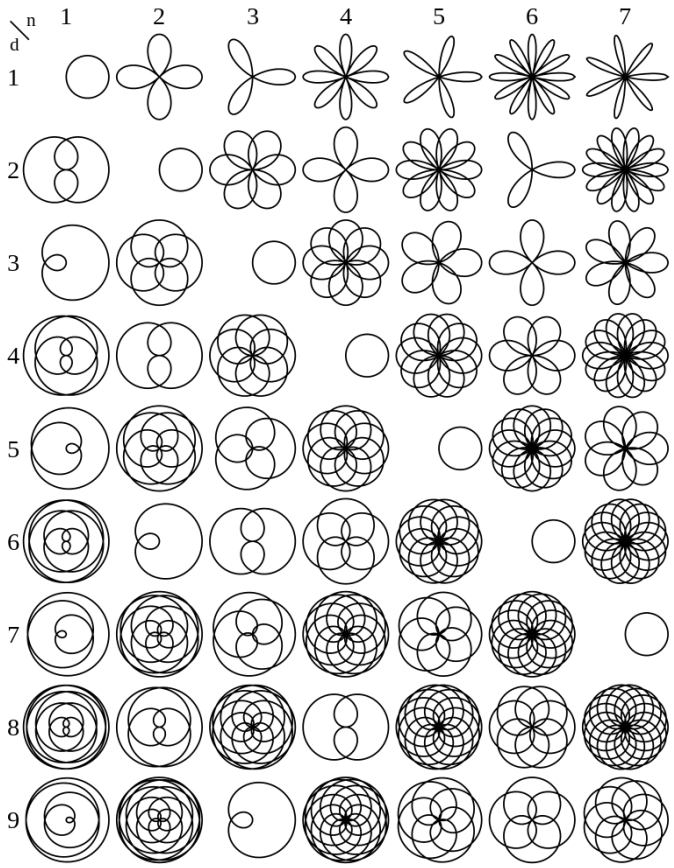
Nakon što korisnik iz padajućeg popisa izabere vrstu grafike koju želi evoluirati algoritam generira početnu populaciju od osam grafika s nasumičnim parametrima. Svakom sljedećom generacijom pomoću istog koda za generiranje grafike genetski algoritam generira novih osam grafika (parametri su evoluirani geni). Programski kod za generiranje grafike se ne mijenja već je on kroz cijeli proces načelno isti. Iako se sam kod ne mijenja, ovisno o vrijednostima dobivenih parametara i kompleksnosti algoritma za generiranje grafike - generirane grafike mogu biti raznolike i do neprepoznatljivosti.

2.4.1. Rozete

Rozeta, ruža ili krivulja *rhodonea* jedan je od primjera kako se s jednom formulom može postići niz vizuala. Izraz koji opisuje krivulju glasi [9]:

$$r = \cos(k\theta)$$
$$x = \cos(k\theta)\ \cos(\theta)$$
$$y = \cos(k\theta)\ sin(\theta)$$

S obzirom na vrijednost *k* unaprijed su poznate sve kombinacije krivulja koje je moguće postići s navedenim izrazom. (Slika 8)



Slika 8 – Rozete definirane s $r=cos(k*\theta)$ za različite vrijednosti $k=n/d$

(izvor: http://commons.wikimedia.org/wiki/File:Rose-rhodonea-curve-7x9-chart-)

Sada je izraz za dobivanje krivulje potrebno napisati unutar p5.js sintakse definirajući koji su to neodređeni parametri koji će činiti genetski kod grafika koji će se u konačnici biti dio evolucijskog procesa.

Prvo je potrebno odrediti genetski zapis grafike. U ovom slučaju to su varijable: *n*, *d* (iz izraza za krivulju), *scaleNum* (koeficijent za skaliranje), *b1*, *b2*, *b3* (tri komponente boje u RGB sustavu). S obzirom da su geni u svom originalnom zapisu definirani u rasponu 0-1, potrebno im je promijeniti raspon s naredbom *map*.

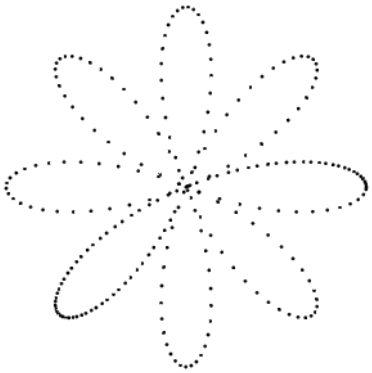
```
function rozeta(){
  let b1 = map(genes[6], 0, 1, 0, 255);   // Novi raspon je: 0-255
  let b2 = map(genes[5], 0, 1, 0, 255);   // Novi raspon je: 0-255
  let b3 = map(genes[4], 0, 1, 0, 255);   // Novi raspon je: 0-255
  let n = floor(map(genes[0], 0, 1, 1, 8));  // Novi raspon je: 1-7
  let d = floor(map(genes[1], 0, 1, 1, 10)); // Novi raspon je: 1-9
  let scaleNum = map(genes[2], 0, 1, 0.2, 1);// Novi raspon je: 0.2-1.0

  push(); // Spremanje prethodnog stila u canvasu
  // Pozivanje funkcije za iscrtavanje
  a_rozeta(n, d, scaleNum, b1, b2, b3);
  pop(); // Vraćanje na prethodno stanje u canvasu
}
```

Isječak koda 7 – Definiranje funkcije za dodjeljivanje gena rozeti

Petlja u funkciji za iscrtavanje (12.-18. redak) određuje točke po kojima će određena rozeta biti ocrtana. (Slika 9)

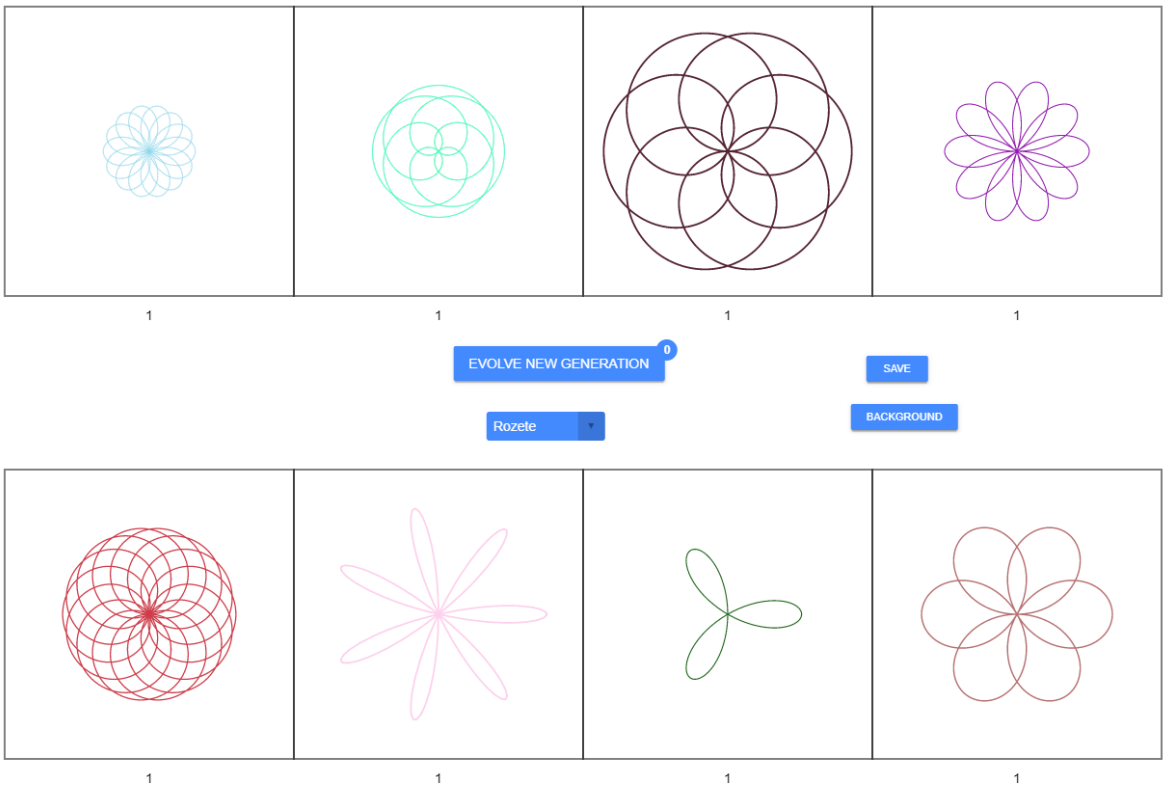
```
function a_rozeta(n, d, scaleNum, b1, b2, b3) {
  colorMode(RGB);    // Postavljanje sustava boje u RGB
  angleMode(RADIANS);// Postavljanje sustava u radijane
  scale(scaleNum);   // Skaliranje grafike ovisno o scaleNum (20%-100%)
  stroke(b1, b2, b3);// Postavljanje obojenja linije
  noFill();          // Uklanjanje boje ispune
  strokeWeight(2);   // Postavljanje debljine crte na 2 pixela

  var k = n / d;     // Definiranje varijable k

  beginShape();      // Početak iscrtavanja lika
  for (var a = 0; a < TWO_PI * d; a += 0.03) {
    var radius = 140 * cos(k * a);
    var x = radius * cos(a);
    var y = radius * sin(a);
    vertex(x, y); // Iscrtava se jedna po jedna točka po putanji krivulje
                  prema zadanom izrazu ovisno o varijablama n, d
  }
  endShape(CLOSE); // Zatvaranje lika određenog točkama iz petlje iznad
}
```

Isječak koda 8 – Definiranje funkcije za iscrtavanje rozete



Slika 9 – Putanja krivulje određena točkama prije iscrtavanja

Nakon što je algoritam za generiranje rozeta određen genetski algoritam generira grafike:



Slika 10 – Početna populacija rozeti

Ovaj algoritam ne daje naročito zanimljive grafike s obzirom da su sve kombinacije rozeti koje se mogu postići unaprijed poznate (Slika 8). Osim gena *n* i *d* koji određuju vrstu rozete, u genetskom zapisu nalaze se još 4 gena no oni utječu samo na veličinu i boju rozete.

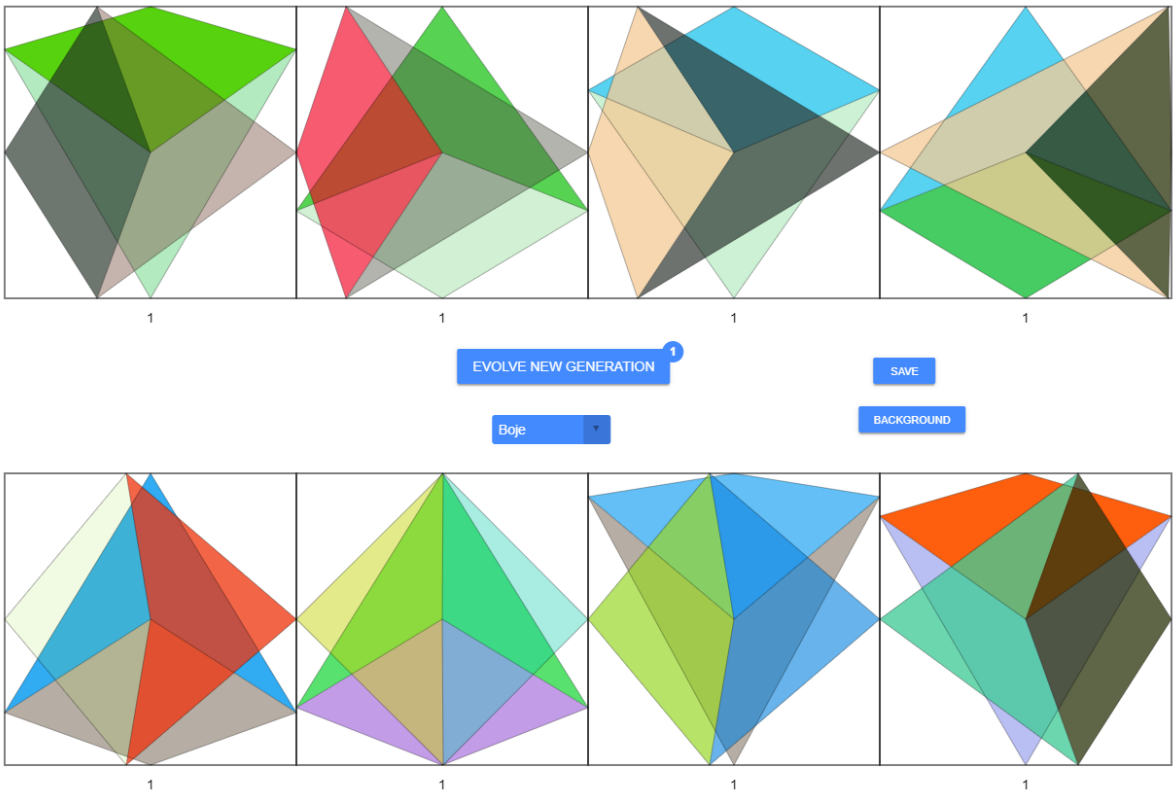
2.4.2. Miješanje boja poligonima

Ovaj primjer generirane grafike sastoji se od četiri mnogokuta (poligona), svi mnogokuti dijele jednu zajedničku točku u sredini grafike. Mnogokuti su povezani u parove tako da četiri mnogokuta tvori dva veća. Oba velika mnogokuta imaju 5 točaka, jednu na svakom rubu okvira grafike i petu u sredini. Za oba mnogokuta dvije točke na rubu su statične, a dvije pomične (točke su na suprotnim osima za jedan i drugi mnogokut).

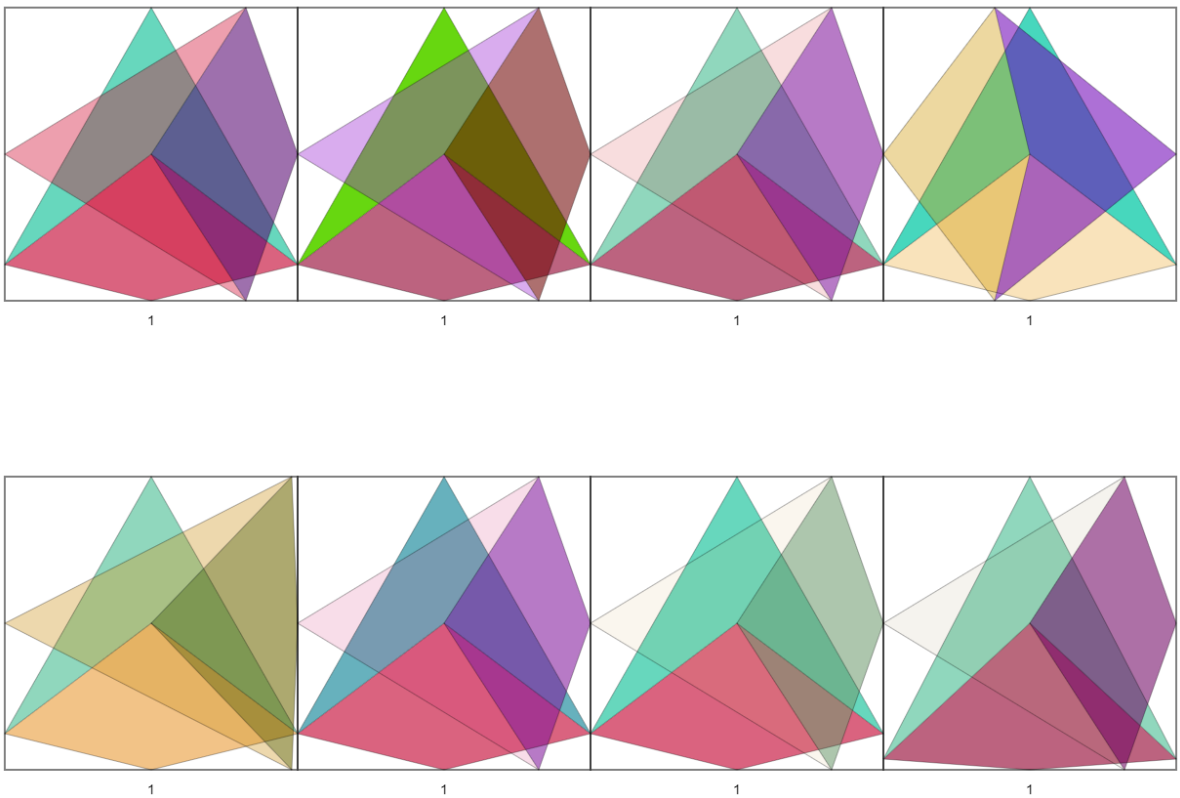
Ideja grafike je da mnogokuti tvore različitu podjelu prostora dok svaki od četiri manja mnogokuta ima nasumičnu boju ispune. Uz nasumičnu boju, ispuna mnogokuta sadrži i nasumičnu vrijednost prozirnosti što znači da će svi mnogokuti biti djelomično transparenti. Preklapanje tako obojenih mnogokuta rezultira površinama s novim bojama ili nijansama boje.

Algoritam za generiranje ove grafike koristi 18 gena za genetski zapis. 16 gena služi za definiranje obojenja: 4 gena za određivanje RGB komponenti i prozirnosti za svaki od 4 mnogokuta, te 2 gena za određivanje pozicije na osima za dva veća mnogokuta. Genetski zapis definiran je na način kao u prošlom primjeru.

Slika 11 – Primjer grafike mnogokuta



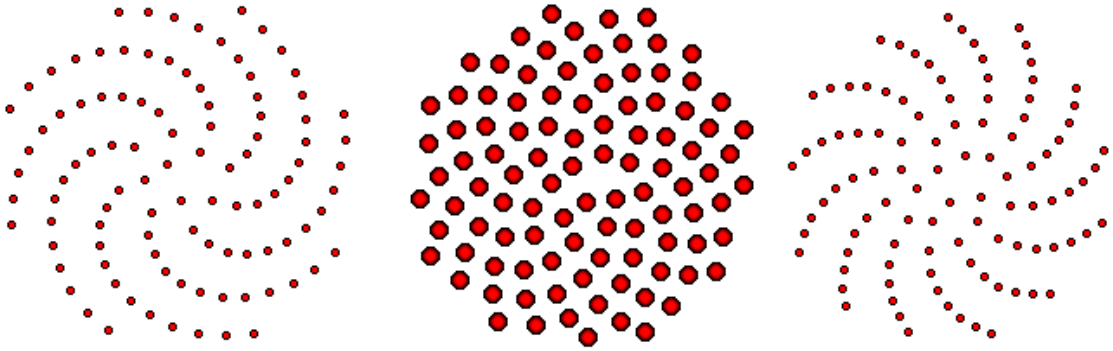
Slika 12 – Generirane grafike na početku evolucijskog procesa (prva generacija)



Slika 13 – Generirane grafike na usred evolucijskog procesa (deseta generacija)

2.4.3. Fermatova spirala

U sredini suncokreta i tratinčice pojavljuje se mreža spirala u Fibonaccijevom nizu jer se divergencija približava zlatnom rezu. Oblik spirale ovisi o rastu elemenata koji se uzastopno generiraju. Fermatova spirala je parabolička spirala, a model te spirale koji će aplikacija generirati predložio je H.Vogel 1979. godine. [10]



Slika 14 – Uzorak cvjetova prema Vogelovom modelu

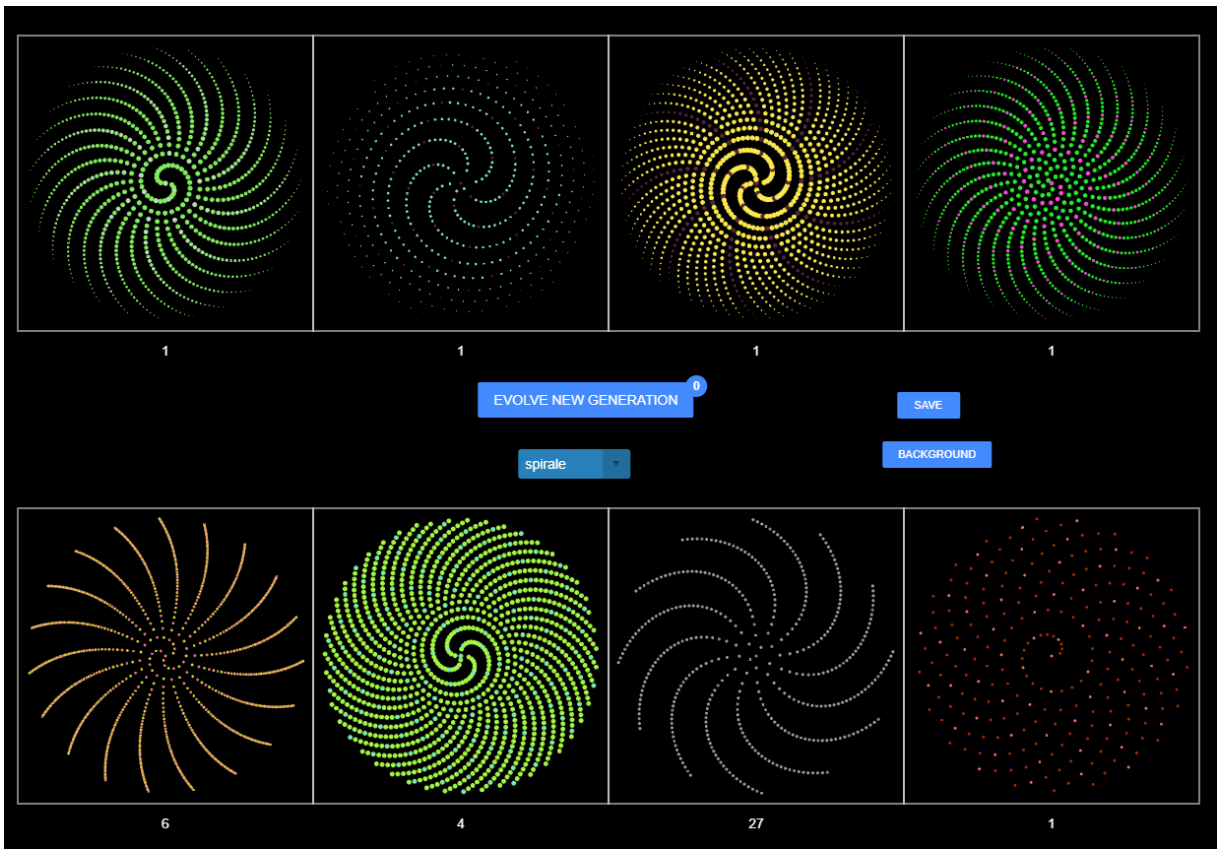
(izvor: https://en.wikipedia.org/wiki/Fermat%27s_spiral#/media/File:Sunflower_spiral.png)

Sljedeći algoritam generira Fermatovu spiralu. Ovisno o koeficijentu rotacije (*turnFraction*) petlja generira različite spirale od točaka. Određeno je 50% šanse da su grafike u početnoj populaciji sastavljene od točki koje se kontinuirano smanjuju (13. – 15. redak), u suprotnom je veličina točaka konstantna vrijednost. Točke su uvijek ispunjene s dvije boje, no ovisno o vrijednosti gena (*highlight*) postoji sedam mogućnosti: prvotno su sve točke u primarnoj boji, a sekundarna boja može biti primijenjena na svaku drugu, treću, četvrtu, petu,… sve do svake osme točke u spiralnom nizu.

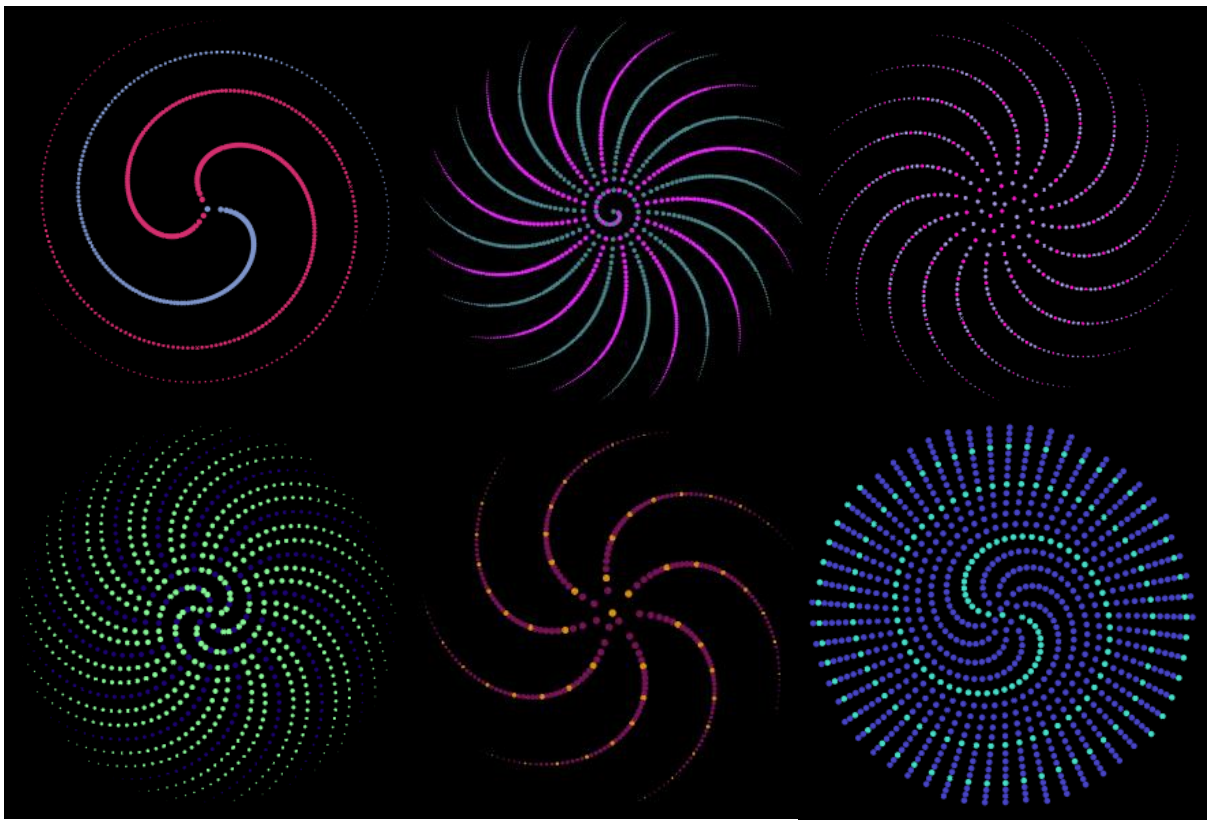
Genetski zapis za generiranje ove grafike sastoji se od 11 gena: 6 komponenti RGB boje (3 za primarnu boju i 3 za sekundarnu boju), broj točaka (*numPoints*) koji se koristi za iscrtavanje spirale (gustoću spirale), koeficijent rotacije (*turnFraction*) koji određuje vizualni izgled spirale, redni broj točke u spiralnom nizu (*highlight*) koja će kontinuirano biti bojana sekundarnom bojom, početna veličina (radijus) točke (*startSize*), te nasumična vrijednost (*resizeChance*) koja određuje hoće li se veličina točke postupno smanjivati ili ne.

```
// Petlja se ponavlja za određeni broj točaka (numPoints)
for (var i = 0; i < numPoints; i++) {
  // Udaljenost u rasponu 0-1 tijekom cijele petlje
  var dst = i / (numPoints - 1);
  // Kut rotacije za djelić kružnice
  var angle = 2 * PI * turnFraction * i;
  // Određivanje koordinata točke
  var x = dst * cos(angle);
  var y = dst * sin(angle);

  // Određivanje veličine točke
  var size = startSize;
  if (resizeChance < 0.5) {
     size = startSize - map(i, 0, numPoints, 0.05, startSize);
  }
  // Određivanje obojenja točke
  if ((i + highlightOffset) % highlight == 0) {
    stroke(color_b);
    fill(color_b);
  }
  else{
    stroke(color_a);
    fill(color_a);
  }
  // Iscrtavanje točke
  circle(x, y, size);
}
```

Isječak koda 9 – Petlja za iscrtavanje točaka spirale



Slika 15 – Početna populacija generiranih spirala



Slika 16 – Generirane spirale

3. ZAKLJUČAK

Upotrebom genetskog algoritma uspješno je razvijena web aplikacija za navođenu izradu generativnih grafika. Genetski algoritam se umjesto klasične selekcije koristi interaktivnom selekcijom što u isto vrijeme olakšava razvoj aplikacije, ali i smanjuje brzinu evolucijskog razvoja grafika u procesu. Interaktivna selekcija olakšava razvoj aplikacije jer genetski algoritam u tom slučaju ne koristi klasičnu *fitness* funkciju kao sustav vrednovanja generiranih rezultata već je korisnik aplikacije ključan element potreban za vizualno vrednovanje generiranih grafika. S druge strane, čovjek u procesu usporava genetski algoritam jer zamjenjuje računalo koje bi u slučaju da se koristi *fitness funkcija* neopisivo brže ocjenjivalo generirane populacije rješenja te samim tim bi i evolucijski napredak bio značajno brži.

Također, osim procesa selekcije, ključan dio genetskog algoritma svodi se na upravljanje postavkama genetskog algoritma. S obzirom na problem koji se rješava potrebno je prije početka rada algoritma odrediti vrijednosti postavki kao što su: veličina populacije (broj elemenata u populaciji) i vjerojatnost mutacije. Uvećani broj elemenata u populaciji može u isto vrijeme usporiti sam proces evolucijskog napretka, ali i povećati vjerojatnost da se dođe do željenog rješenja. Vjerojatnost mutacije mora biti optimalna vrijednost koja najviše ovisi o veličini populacije i vrsti problema za koji se pokušava naći rješenje. Genetski algoritam s premalom ili prevelikom vjerojatnosti mutacije proizvest će rješenje koje nije ni približno onom idealnom.

Web aplikacija za navođeno generiranje jednostavnih grafika nalazi se na adresi: http://benjamin-cacan.from.hr/ga/

4. LITERATURA

1. Genetic Algorithms and Evolutionary Computation, http://www.talkorigins.org/faqs/genalg/genalg.html, 2. 8. 2019.
2. D. Shiffman, Nature of Code, 2012.
3. M. Golub, Genetski algoritam: prvi dio, 2004.
4. Galapagos Interactive Exhibit, http://www.karlsims.com/galapagos/, 20.8.2019.
5. p5.js Introduction, http://www.geeksforgeeks.org/p5-js-introduction/, 18.8.2019.
6. I. Senkić, Primjena staničnih automata u generativnoj umjetnosti, diplomski rad, Prirodoslovno-matematički fakultet Sveučilišta u Zagrebu, Zagreb, 2018.
7. About Daniel Shiffman, http://shiffman.net/about/, 15.8.2019.
8. p5.js reference, http://p5js.org/reference/, 25.7.2019.
9. http://en.wikipedia.org/wiki/Rose_(mathematics), 28. 7. 2019.
10. http://en.wikipedia.org/wiki/Fermat%27s_spiral, 15.8.2019.

5. POPIS SLIKA

6. POPIS MANJE POZNATIH RIJEČI

evolucijski proces - proces u kojem nizom promjena ili razvojnih stupnjeva organizam stječe karakteristične značajke

genetski algoritam - metoda optimiranja koja imitira prirodni evolucijski proces

interaktivna selekcija - vrsta selekcije u genetskog algoritmu gdje se *fitness* vrijednost računa na osnovu korisnikove interakcije

križanje - proces prenošenja genetskog materijala, svojstva roditelja prenose se na njihovu djecu

mutacija - proces slučajne promjene gena

prirodna selekcija - neslučajni proces kojim određene biološke značajke odnosno genotipovi nestaju iz populacije tijekom vremena kao posljedica slabije prilagođenosti uvjetima života

reprodukcija - proces razmnožavanja pomoću križanja (novonastale jedinke nasljeđuju obilježja svojih roditelja)

selekcija - proces izdvajanja najsposobnijih jedinki

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET
Getaldićeva 2
Zagreb, 16. 9. 2019.

Temeljem podnijetog zahtjeva za prijavu teme završnog rada izdaje se

R J E Š E N J E

kojim se studentu/ici Benjaminu Cacanu, JMBAG 0128059469, sukladno čl. 5. st. 5. Pravilnika o izradi i obrani završnog rada od 13.02.2012. godine, odobrava izrada završnog rada, pod naslovom: Razvoj aplikacije za navođeno generiranje jednostavnih grafika uporabom genetskog algoritma, pod mentorstvom doc. dr. sc. Tibora Skale.

Sukladno čl. 9. st. 1. Pravilnika o izradi i obrani završnog rada od 13.02.2012. godine, Povjerenstvo za nastavu, završne i diplomske ispite predložilo je ispitno Povjerenstvo kako slijedi:

1. doc. dr. sc. Stanić Loknar Nikolina, predsjednik/ica
2. doc. dr. sc. Skala Tibor, mentor/ica
3. doc. dr. sc. Rudolf Maja, član/ica

Dekan

Prof. dr. sc. Nikola Mrvac