Web aplikacija - generator postera

Kosi, Ozren

Sveučilište
Sjever

Završni rad br. 686/MM/2020

Web aplikacija - generator postera

Ozren Kosi, 2713/336

Varaždin, rujan 2020.

Sveučilište
Sjever

Multimedija, oblikovanje i primjena

Završni rad br. 686/MM/2020

Web aplikacija - generator postera

Student

Ozren Kosi, 2713/336

Mentor

mr. sc. Vladimir Stanisavljević, viši predavač

Varaždin, rujan 2020.

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Ozren Kosi | MATIČNI BROJ 2713/336

DATUM 03.09.2020. | KOLEGIJ Programski alati 3

NASLOV RADA Web aplikacija - generator postera

NASLOV RADA NA ENGL. JEZIKU Web application for generating posters

MENTOR mr.sc. Vladimir Stanisavljević | ZVANJE Viši predavač

ČLANOVI POVJERENSTVA

1. doc.dr.sc. Dean Valdec - predsjednik
2. doc.dr.sc. Andrija Bernik, pred. - član
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. pred. Snježana Ivančić Valenko, dipl.graf.ing. - rezervni član
5.

Zadatak završnog rada

BROJ 686/MM/2020

OPIS

Processing je programerska platforma kojoj je osnovni cilj olakšati programiranje za početnike za primjene bliske multimedijalnim umjetnicima te drugim ne-programerskim zanimanjima. Pomoću njega se jednostavnije nego u drugim popularnim programskim jezicima može se ostvariti grafički prikaz i interakcija sa okolinom što ga čini pogodnim za primjene u multimedijskim programima. Glavna inačica sastoji se od programskog jezika temeljenog na Java-i te razvojnog okruženja, ali postoji i JavaScript inačica p5.js.
U radu je potrebno:
* ukratko prikazati povijest, razvoj, mogućnosti i primjere korištenja platforme Processing s naglaskom na p5.js,
* pomoću p5.js izvesti jednostavnije i složene samostalne primjere i opisati korake i primitive korištene u izradi,
* osmisliti i izraditi web stranicu za sintetiziranje grafičkog proizvoda poput plakata
Detaljno opisati sve korištene tehnologije i korake u radu potrebne da bi se ostvarilo traženo te detaljno opisati stečena iskustva i postignute rezultate.

ZADATAK URUČEN 17. 09. 2020. | POTPIS MENTORA

REPUBLIKA HRVATSKA SVEUČILIŠTE SJEVER KOPRIVNICA

Sažetak

Generator postera je web-aplikacija izrađena pomoću alata p5.js, JavaScript biblioteke i skupa alata koji olakšavaju upotrebu programskog jezika JavaScript za kreativno kodiranje.

Koncept web-aplikacije je algoritmičko generiranje dizajna postera prema zadanim parametrima pritom koristeći Perlinov šum. Korisnik ima opcije odabrati boju pozadine postera, oblik na kojem će se bazirati dizajn, broj pojavljivanja odabranog oblika te eksportiranje u JPG rasterski format ili SVG vektorski format. Eksportirani poster korisnik može dalje uređivati u programima za grafičku obradu.

Aplikaciji je moguće pristupiti na web-adresi: http://arwen.unin.hr/~ozkosi/zavrsni-rad/index.html.

Ključne riječi: Generator postera, web-aplikacija, Perlinov šum, p5.js, JavaScript

Summary

Poster generator is a web application created using the p5.js, JavaScript library, and a set of tools that make creative coding simple and accessible for everyone.

The concept of the web application is to algorithmically generate a poster design according to given parameters using Perlin noise. The user has options to select the background color of the poster, elements on which the design will be based, the number of occurrences of the selected design element and export to a JPG raster format or SVG vector format. The user can further edit the exported poster in graphics processing programs.

The web application can be accessed at a web address: http://arwen.unin.hr/~ozkosi/zavrsni-rad/index.html.

Keywords: Poster generator, web application, Perlin noise, p5.js, JavaScript

Sadržaj

1. Uvod

U današnjoj eri digitalizacije sve se više i više stavlja naglasak na automatizaciju procesa. U proizvodnoj industriji automatizira se proces sklapanja proizvoda. Poslove za koje je prije desetak godina trebalo stotinu para ruku danas obavlja par robota.

U kućanstva su ušli glasovni asistenti kao što su Amazon Alexa ili Apple Siri koji automatiziraju radnje kao što su podizanje roleta, reguliranje termostata ili kontrolu osvjetljenja.

U kreativnoj industriji programi kao što je Adobe Photoshop dobili su mogućnost automatskog prepoznavanja i selektiranja subjekta na slici, time ubrzavajući dugotrajan i mukotrpni proces. Pogotovo je olakšano selektiranje kose, koju je ručnom tehnikom selektiranja gotovo nemoguće savršeno selektirati.

Automatizacija nam pomaže da energiju koju bismo u protivnome ulagali u repetitivne procese uložimo na kreativno razmišljanje o apstraktnijim konceptima.

Potaknut idejom odlučio sam napraviti web-aplikaciju koja će na klik generirati dizajn postera koji je moguće eksportirati u popularnim slikovnim formatima te ga dalje uređivati u programima za uređivanje vektorske grafike kao što je Adobe Illustrator.

Aplikaciju ću napraviti korištenjem kombinacija HTML-a, CSS-a i JavaScripta, preciznije JavaScript biblioteke p5.js.

2. Tehnologije korištene u izradi web-aplikacije

Prilikom izrade web-aplikacije koristio sam različite tehnologije koje ću u nastavku pobliže opisati.

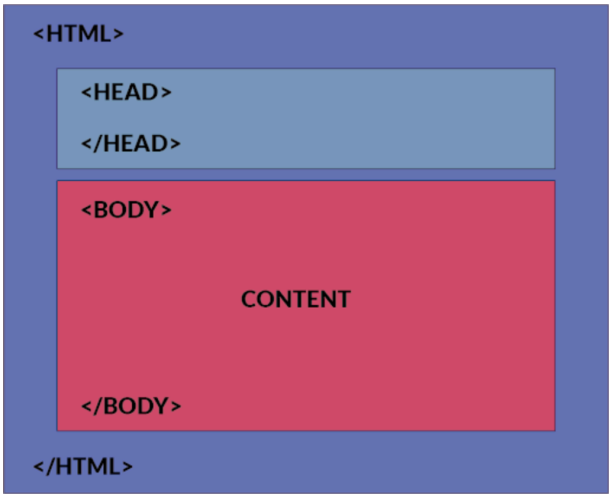
2.1. HTML

HTML **[1][2]** (eng. Hyper Text Markup Language) je jezik kojim opisujemo izgled web stranica pomoću HTML oznaka (eng. tag). Web stranica se sastoji od raznih elemenata kao što su: tekst, slike, tablice i slično.

Definirao ga je 1990. godine sir Timothy Berners-Lee **[3]**, danas ravnatelj World Wide Web konzorcija (W3C, https://www.w3.org), organizacije koja brine o standardizaciji web tehnologija i razvoju weba.

HTML oznake u tekstualnom obliku određuju položaj i način prikazivanja elemenata web stranice. Web preglednik interpretira (tumači) HTML oznake i stvara prikaz web stranice. To je razlog zašto se prikaz web stranice u različitim web preglednicima može katkad razlikovati. HTML oznake možemo razumjeti kao naredbe web pregledniku koje određuju kako će se prikazati sadržaj web stranice. Svaku HTML oznaku navodimo unutar znakova < > , odnosno izlomljenih zagrada. Primjerice, oznaka kojom počinje HTML dokument je <html>.

Većina HTML oznaka dolazi u paru, odnosno postoji početna i završna HTML oznaka. Primjerice, oznaka <html> označava početak HTML dokumenta, a oznaka </html> označava završetak HTML dokumenta. Vidimo da završna oznaka ispred svojeg naziva ima desnu kosu crtu /.



Slika 1 - Vizualna reprezentacija strukture HTML dokumenta (izvor: http://estmary.2020.madeatep-s.org/html-structure/, pristupljeno 4. rujna 2020.)

HTML dokument započinje oznakom <html> i u osnovi je podijeljen na dva dijela HTML oznaka: zaglavlje i tijelo. Zaglavlje sadržava osnovne informacije o naslovu i obilježjima stranice. Oznaka za početak zaglavlja je <head>, a za završetak zaglavlja </head>. Tijelo sadržava oblikovani sadržaj, a njegova početna oznaka je <body>, a završna </body>.

Da bismo kreirali najjednostavniju web stranicu potrebna su nam još dva para oznaka, oznaka naslova i oznaka odlomka. Oznaka naslova piše se unutar oznaka za zaglavlje. Početna oznaka naslova je <title>, a završna </title>. Unutar ovih oznaka upisuje se naslov web stranice.

Početna oznaka odlomka je , a završna . Oznake odlomka pišemo unutar oznaka za tijelo HTML dokumenta.

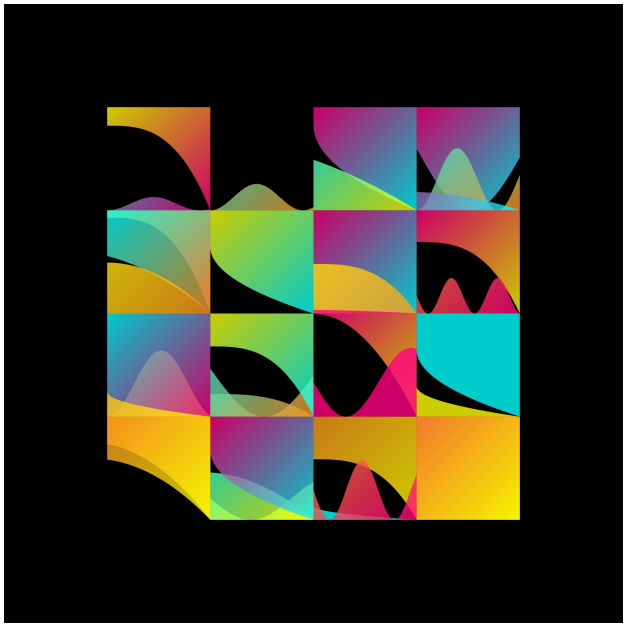
HTML se konstantno ažurira s novim oznakama. Trenutna najnovija verzija je HTML5.

2.1.1. HTML5 Canvas

HTML5 uključuje novi element zvan HTML5 Canvas **[4] [5]**. Koristi se za razvoj dinamične grafike, online i izvanmrežnih igara, animacija, interaktivnih videa i audia.

Canvas element jednostavno pretvara običnu web stranicu u dinamičnu web aplikaciju ili čak u mobilnu aplikaciju za upotrebu na pametnim telefonima i tabletima.

Za crtanje na Canvasu potrebno je koristiti programske jezike JavaScript te ponekad CSS.



Slika 2 - Primjer grafike nacrtane HTML5 Canvasom (izvor: https://i.redd.it/3w6a3175ibb11.png, pristupljeno 4. rujna 2020.)

2.2. CSS

CSS **[6] [7]** je jezik koji služi za oblikovanje web-stranica. Uz HTML, jezik pomoću kojeg se definiraju struktura i sadržaj web-stranica, CSS je osnovna tehnologija na kojoj se temelji današnji web.

CSS je kratica za Cascading Style Sheets. Pojam style sheet često se upotrebljava za datoteku koja sadrži CSS-kôd. Dakle, style sheet je datoteka koja definira stil, odnosno izgled web-stranice. Riječ cascading označava kaskadnu primjenu CSS-pravila. CSS-pravilo može se napisati tako da bude primijenjeno na sve elemente ili samo na neke elemente ili tako da vrijedi samo za točno određeni element.

Prije pojave CSS-a oblikovanje izgleda web-stranice do određene razine bilo je moguće postići i u HTML-u. No, time se stvorio problem miješanja sadržaja i strukture s kôdom čija je jedina svrha bila prezentacija. HTML-kôd za definiranje izgleda morao se ponavljati iznova na svakom elementu i na svakoj stranici u web-sjedištu.

Pojavom CSS-a nastoji se riješiti taj problem. Glavna je ideja CSS-a odvajanje prezentacijskog kôda u zasebne datoteke i njegovo definiranje pomoću jednostavnih pravila koja se mogu odnositi na više elemenata odjednom.

Prva verzija CSS-a definirana je krajem 1996. No do usvajanja CSS-a od strane autora web-sadržaja i proizvođača web-preglednika prošlo je još dosta vremena. Vrlo dugo web-preglednici nisu dosljedno implementirali CSS-specifikaciju pa se autori nisu mogli pouzdati da će stranice izgledati približno isto u svim preglednicima. Razvijeni su brojni trikovi u CSS-u čija je namjena bila da isprave neočekivana ponašanja u nekim preglednicima. Današnja situacija je po tom pitanju puno bolja, iako se i danas savjetuje provjera izgleda stranice u što više preglednika.

Trenutačna verzija CSS-a je CSS 3, koji je donio brojne novitete i poboljšanja. Razvojem CSS-a bavi se World Wide Web Consortium (W3C), tijelo zaduženo za razvoj web-standarda u suradnji s proizvođačima preglednika i zainteresiranim web-autorima.

Style sheet u CSS-u sastoji se od nekoliko pravila. Svako pravilo sastoji se od selektora i deklaracijskog bloka.

Svako pravilo moramo započeti **selektorom**, a najjednostavniji selektor je upravo naziv elementa (u ovom slučaju odabiremo HTML element paragraph):

```
p {

}
```

Unutar vitičastih zagrada najprije se navodi **svojstvo** koje se postavlja. U ovom primjeru radi se o boji teksta (CSS-svojstvo color):

```
p {
      color:
}
```

Nakon što se navede svojstvo koje se želi postaviti, dolazi dvotočka, a iza nje **vrijednost** na koju se postavlja to svojstvo (u ovom primjeru radi se o nazivu boje):

```
p {
      color: red;
}
```

Svojstvo i vrijednost zajedno čine **deklaraciju**.

2.3. JavaScript

JavaScript **[8] [9]** je skriptni jezik kojim se u statičke HTML-stranice mogu uvesti interaktivni elementi. JavaScript omogućuje izvršavanje određenih radnji u inače statičnim HTML-dokumentima, npr. interakciju s korisnikom, promjenu svojstava preglednikova prozora ili dinamičko stvaranje HTML-sadržaja.

JavaScript se razvija od 1995. godine kada je Netscape objavio nekoliko prvih inačica jezika.

JavaScript je interpreter, što znači da se skripta izvršava odmah naredbu po naredbu, bez prethodnog prevođenja (kompiliranja) cijelog programa i kreiranja izvršne datoteke.

Neke od funkcija koje omogućava JavaScript su:

- Reagiranje na događaje – moguće je postaviti da se skripta izvršava kada se dogodi neki događaj, npr. kada se stranica učita, ili kada korisnik klikne na određeno dugme ili drugi HTML element
- Čitanje i pisanje HTML elemenata - JavaScript može pročitati i promijeniti sadržaj nekog HTML elementa
- Pretvaranje teksta u HTML kod
- Validiranje (provjeru ispravnosti i vjerodostojnosti) podataka - JavaScript može validirati podatke prije nego se pošalju na server, čime se server oslobađa dodatne obrade
- Detektiranje preglednika kojeg korisnik upotrebljava – na osnovu tog prepoznavanja preglednika JavaScript može učitati drugačiju stranicu ovisno o pregledniku tako da se učita stranica koja je posebno dizajnirana za taj preglednik
- Kreiranje "kolačića" (cookies) - JavaScript može pohraniti i učitati informacije o korisnikovom računalu

Funkcionalnost JavaScripta se može proširiti korištenjem "biblioteka" (libraryja). U ovom projektu koristio sam biblioteku p5.js.

2.3.1. p5.js biblioteka

p5.js **[10] [11]** (https://p5js.org) je biblioteka i skup alata koji olakšavaju upotrebu programskog jezika JavaScript za kreativno kodiranje. Temelji se na Processingu, kreativnom okruženju za kodiranje koje su izvorno razvili Ben Fry i Casey Reas. Jedan od glavnih ciljeva Processinga i p5.js-a je početnicima olakšati učenje programiranja interaktivnih, grafičkih aplikacija.

Prednost upotrebe JavaScript programskog jezika je njegova široka dostupnost i sveprisutna podrška: svaki web preglednik ima ugrađen JavaScript tumač, što znači da se p5.js programi mogu pokretati u bilo kojem web pregledniku. JavaScript je također definiran međunarodnim standardom te su većina JavaScript tumača otvorenog koda i slobodno dostupni. Nitko ne posjeduje JavaScript i svatko ga može besplatno koristiti.

Osim što je sjajan alat za učenje kodiranja, p5.js ima i druge zanimljive primjene, poput vizualizacije podataka ili generativnog dizajna. Postoje i brojne knjižnice koje proširuju osnovnu funkcionalnost p5.js-a, npr. omogućavanjem DOM interakcija, integriranjem hardvera poput web kamera ili mikrofona i vizualizacijom geo-podataka.

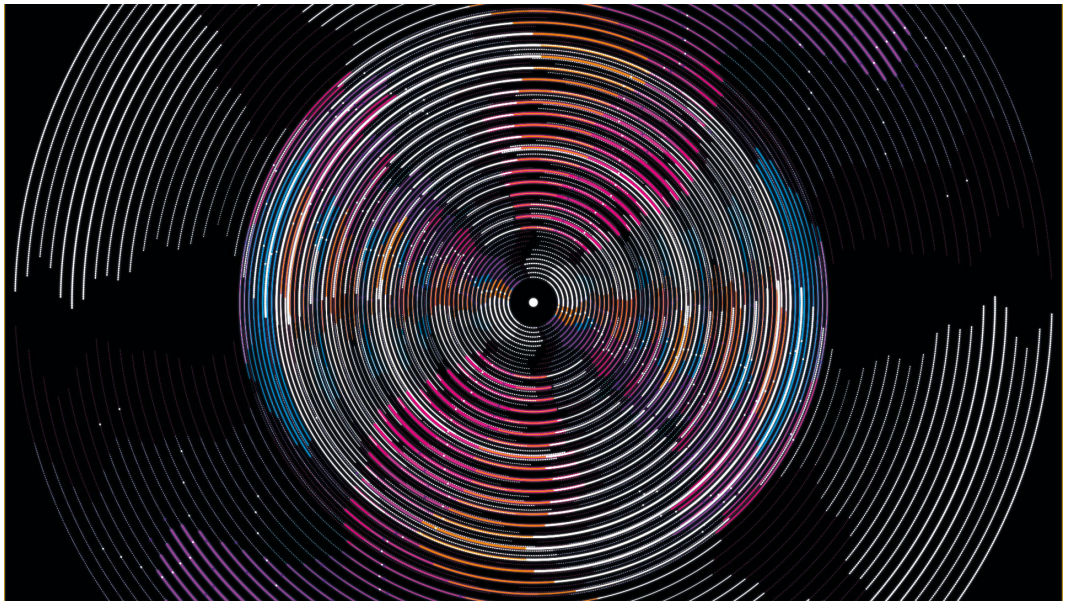
2.3.2. Platforma za kreativno programiranje - Processing

Processing **[12] [13] [14]** (https://processing.org) je grafička knjižnica otvorenog koda i integrirano razvojno okruženje (IDE) izgrađeno za elektroničku umjetnost, umjetnost novih medija, s ciljem podučavanja ne-programera osnova računalnog programiranja u vizualnom kontekstu.

Processing koristi jezik Java uz dodatna pojednostavljenja kao što su dodatne klase i aliasirane matematičke funkcije i operacije. Također pruža grafičko korisničko sučelje za pojednostavljivanje faze kompajliranja i izvršenja programa.

Processing je prethodnik drugim projektima uključujući Arduino, Wiring i p5.js.

Projekt su 2001. godine pokrenuli Casey Reas i Ben Fry, obojica iz Grupe za estetiku i računarstvo u MIT Media Labu. 2012. pokrenuli su Processing zakladu zajedno s Danielom Shiffmanom, koji se pridružio kao treći voditelj projekta.



Slika 3 - Primjer vizualizacije zvuka pomoću web inačice Processinga, p5.js (izvor:https://www.creativebloq.com/how-to/data-visualisation-with-p5js, pristupljeno 6. rujna 2020.)

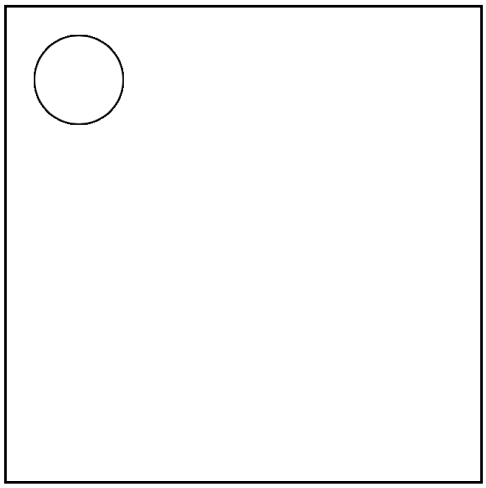
2.3.3. Osnove korištenja p5.js biblioteke

U nastavku ću objasniti najjednostavniji p5.js program.

p5js-osnove.js

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(255);
  ellipse(50, 50, 60, 60);
}
```

REZULTAT



createCanvas(400, 400); je primjer poziva funkcije. p5.js dolazi s nekoliko desetaka ugrađenih funkcija koje obavljaju različite zadatke, poput crtanja oblika na zaslonu ili izračunavanja vrijednosti pomoću matematičkih formula. Učenje p5.js programiranja uglavnom se svodi na učenje ovih naredbi i onoga što rade. Funkcija createCanvas kreira HTML Canvas dimenzija 400 x 400 piksela.

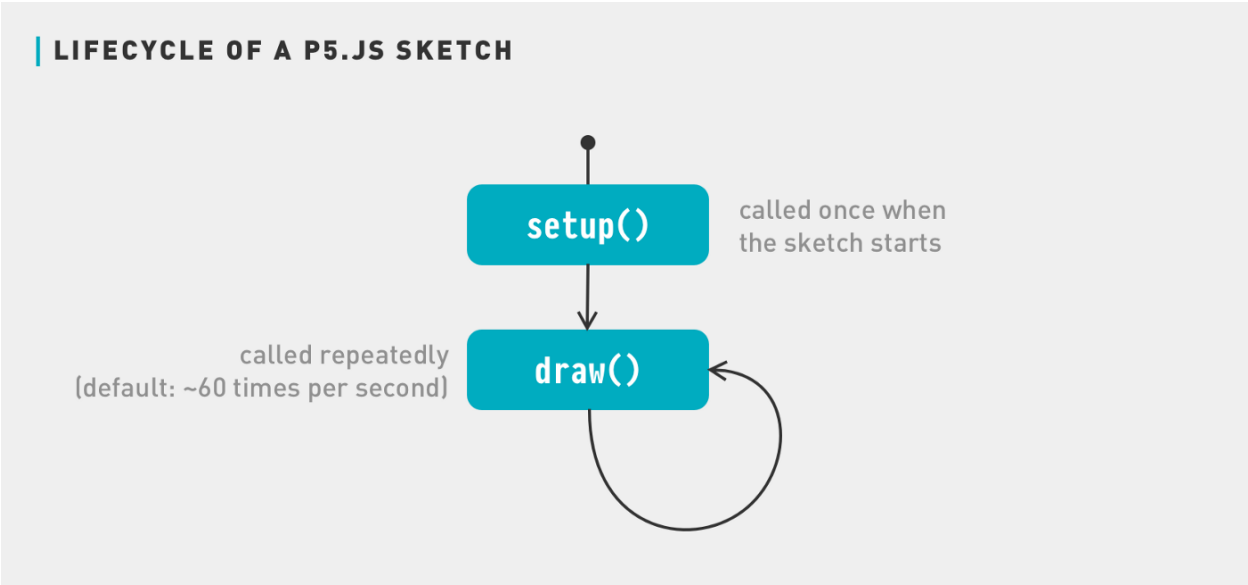
background(255); je funkcija koja crta boju pozadine. U ovom slučaju boja pozadine ima HEX kod 255 koji označuje bijelu boju.

ellipse(50, 50, 60, 60); je funkcija koja crta elipsu na zaslonu na koordinatama x:50px y:50px, širine 60px i visine 60px. Brojke unutar funkcije nazivaju se parametri funkcije. Svaka funkcija završava s točkom i zarezom.

Funkcije **ellipse()** i **background()** napisane su unutar funkcije **draw().** Funkcije **setup()** i **draw()** su osnove p5.js-a.

Funkcija **setup()** poziva se jednom kad se pokrene program. Koristi se za definiranje početnih svojstava okoline kao što su veličina zaslona i boja pozadine te za umetanje medija poput slika i fontova.

Funkcija **draw()** automatski se poziva 60 puta u sekundi tijekom odvijanja programa. Najčešće se koristi za crtanje oblika kao što je u ovom slučaju elipsa.



Slika 4 - Slijed izvršavanja p5.js programa (izvor: https://medium.com/comsystoreply/introduction-to-p5-js-9a7da09f20aa, pristupljeno 4. rujna 2020.)

2.3.4. Napredniji primjer p5.js biblioteke - fraktalno stablo

Za početak potrebno je razjasniti što je fraktal. On je krivulja ili geometrijska figura, čiji svaki dio ima ista obilježja kao i cjelina. Njega je programski najlakše implementirati koristeći rekurziju. Programu dajemo samo par uputa koje referenciraju same sebe te izvršavaju velik broj operacija.

Program započinjemo kreiranjem HTML Canvasa dimenzija 800 x 600 piksela u setup() funkciji naredbom createCanvas(800, 600).

Zatim u draw() funkciji crtamo crnu pozadinu pomoću funkcije background(0).

p5js-fraktalno-stablo.js

```
function setup() {
  createCanvas(800, 600);
}

function draw() {
  background(0);
}
```

Sljedeći korak je premještanje početne koordinate (0, 0) s gornjeg-lijevog kuta na donji-centar. To je početna točka crtanja stabla i time ćemo si olakšati daljnje programiranje. Taj postupak nam omogućava funkcija **translate()**. U našem slučaju ona glasi translate(width / 2, height).

Sad kad se nalazimo na pravilnoj lokaciji možemo početi s crtanjem stabla. Prvu granu duljine 150 piksela crtamo pomoću funkcije **line(0,0,0, -150)** uz pomoć funkcije **stroke()** koja određuje boju linije preko HEX koda.

Time smo nacrtali jednu granu na dnu ekrana. Budući da želimo crtati još grana naredbe za crtanje linije grupirat ćemo u zasebnu funkciju **grana()**. Ta ista funkcija primati će jedan argument *duljina* kojim određujemo duljinu grane.

Sve zajedno kod izgleda ovako:

p5js-fraktalno-stablo.js

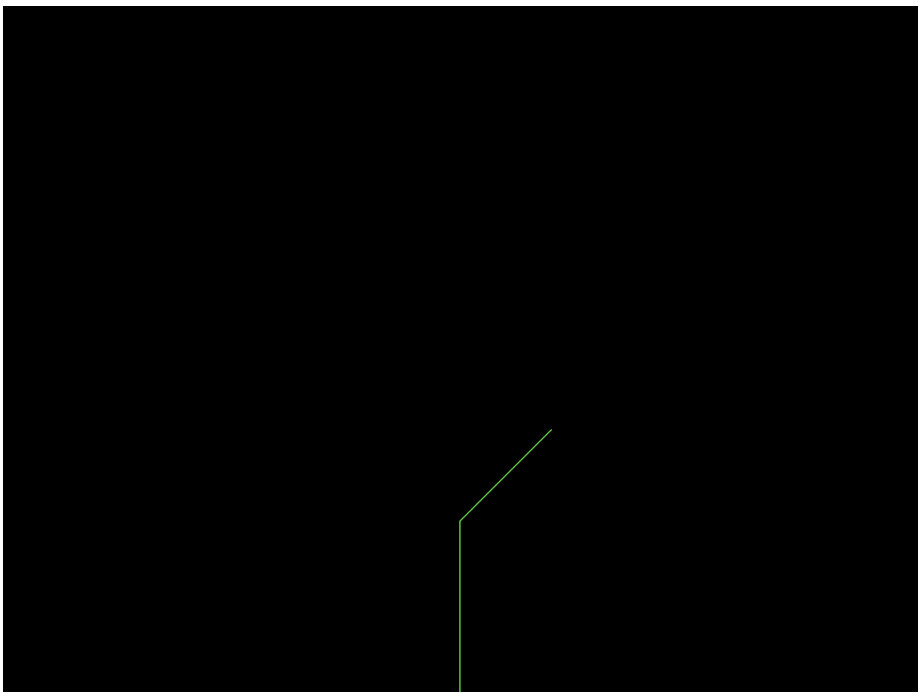
```
function setup() {
  createCanvas(800, 600);
}

function draw() {
  background(0);
  translate(width / 2, height);
  grana(150);
}

function grana(duljina) {
  stroke(50, 50, 250);
  line(0,0,0, -duljina);
}
```

Da bismo nacrtali sljedeću liniju, moramo početi crtati s kraja posljednje crte pod novim kutom. Kut je definiran u odnosu na *PI* pa svojoj funkciji draw() možemo dodati ovo:

p5js-fraktalno-stablo.js

```
function draw() {
  background(0);
  translate(width / 2, height);
  grana(150);
  translate(0, -150);
  rotate(PI / 4);
  grana(150 * 0.75);
}
```

Premjestili smo početnu koordinatu na kraj prve grane. Odatle okrećemo orijentaciju za 45 stupnjeva i crtamo novu granu koja je 3/4 duljine posljednje.



Slika 5 - Dvije tanke, zelene linije koje kreću od dna HTML platna

Sljedeći logičan potez bio bi nacrtati sljedeću granu koja započinje na istom mjestu na kojem je i posljednja. Mogli bismo translatirati na tu poziciju, ali onda, kada napravimo sve više grana, morali bi translatirati gore-dolje svaku račvu što bi postalo vrlo teško za skaliranje.

Taj problem rješavamo rekurzijom i funkcijama **push()** i **pop()**. Te nam funkcije omogućuju crtanje na platnu, a zatim vraćanje na prethodni položaj prije crtanja sljedećeg elementa.

Sljedeći ćemo put okrenuti granu prema suprotnom kutu i nacrtati ju u tom smjeru iste duljine. Funkcija draw() sada izgleda ovako:

p5js-fraktalno-stablo.js

```
function draw() {
  background(0);
  translate(width / 2, height);
  grana(150);
  translate(0, -150);
  push();
    rotate(PI / 4);
    grana(150 * 0.75);
  pop();
```

```
  push();
    rotate(-PI / 4);
    grana(150 * 0.75);
  pop();
}
```

Time smo dobili tri grane. Ali da ponovimo ovu funkciju, još uvijek moramo translatirati niz sljedeću granu, a zatim se vratiti prema dolje i vratiti gore na suprotnu granu.

Da bismo riješili taj repetitivni problem potrebno je koristiti rekurziju.

Rekurzija će nam pomoći da nacrtamo cijelo stablo bez da pišemo svaku liniju koda ručno (što bi bilo i nemoguće). Rekurzija je dovoljno moćna da može slomiti kod pokušavajući ga izvršavati zauvijek ako ne zadamo točku prekida.

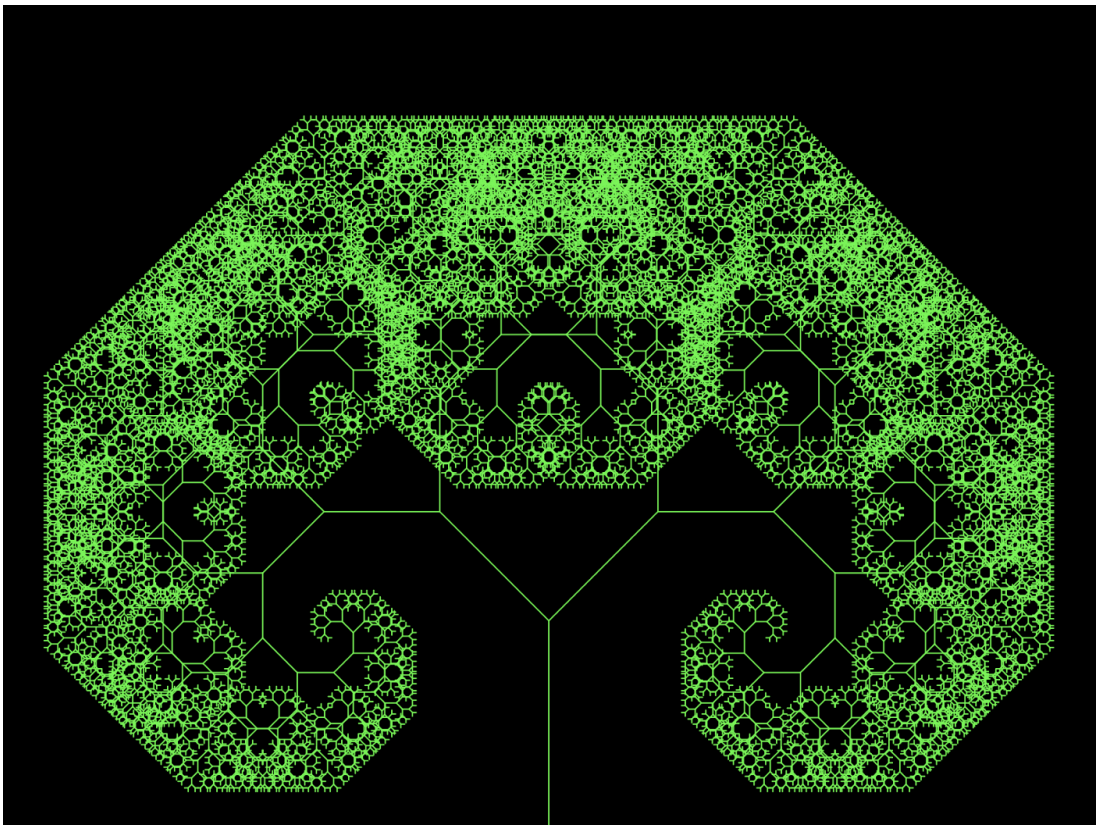
Točku prekida odabrao sam u trenutku kad duljina grane dosegne 1. Premještanjem funkcija push(), pop(), rotate() i grana() unutar funkcije grana(), kod sada izgleda ovako:

p5js-fraktalno-stablo.js

```
function setup() {
  createCanvas(800, 600);
}

function draw() {
  background(0);
  translate(width / 2, height);
  stroke(50, 50, 250);
  grana(150);
  translate(0, -150);
 }

function grana(duljina) {
  line(0,0,0, -duljina);
  translate(0, -duljina);
  if (duljina > 1) {
    push();
      rotate(PI / 4);
```

```
      grana(duljina * 0.75);
    pop();
    push();
      rotate(-PI / 4);
      grana(duljina * 0.75);
    pop();
  }
}
```

Rezultat je fraktalno stablo:



Slika 6 - Fraktalno stablo

Programiranje u p5.js-u se svodi na nadograđivanje jednostavnih koncepata kao što je crtanje oblika i mijenjanje njihove pozicije na ekranu. Koristeći jednostavne funkcije je moguće postići atraktivne animacije pomoću malo linija koda.

Sam p5.js je velika biblioteka koja sadrži naredbe i za kompleksnije programiranje pa ju samim time svatko može koristiti, neovisno o vještini programiranja.

3. Praktični dio

3.1. Povezivanje web-aplikacije s p5.js bibliotekom i izrada potrebnih datoteka

Da bismo počeli programirati p5.js web-aplikaciju najprije je potrebno povezati HTML i JavaScript datoteke te uključiti p5.js biblioteku u projekt.

index.html

```
<!DOCTYPE html>
<html>
<head>
    . . .
    <script src="libraries/p5.js"></script>
    . . .
</head>

<body>
    <script src="main.js"></script>
    . . .
</body>
</html>
```

U zaglavlju HTML-a uključio sam p5.js JavaScript datoteku koju sam skinuo sa službene web-stranice (https://p5js.org/download/) te koju sam prethodno stavio u mapu *libraries*.

Izradio sam novu JavaScript datoteku *main.js* i povezao ju s datotekom *index.html* tijelu HTML-a.

Odmah sam u zaglavlju izradio i uključio CSS datoteku koja će mi kasnije trebati za dizajniranje web-aplikacije, dodao naslov i opis web-aplikacije te dodao *meta* naredbu koja omogućuje prikaz hrvatskih dijakritičkih znakova.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Poster generator</title>
    <meta name="description" content="Generative poster maker using
    Perlin noise made with p5.js">
    . . .
    <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
    . . .
</body>
</html>
```

Zadnja stvar prije početka razvijanja algoritama crtanja bila je izraditi i u tijelu HTML-a povezati dvije JavaScript datoteke *Line.js* i *Shape.js* koji će biti JavaScript objekti koji će sadržavati algoritme za crtanje.

Navedene datoteke sam stavio u mapu *components*.

index.html

```
<!DOCTYPE html>
<html>
<head>
    . . .
</head>

<body>
    <script src="main.js"></script>
    <script src="components/Line.js"></script>
    <script src="components/Shape.js"></script>
</body>
</html>
```

3.2. Izrada vizualne strukture web-aplikacije

Sljedeći korak u izradi web-aplikacije bio je izraditi vizualnu strukturu aplikacije. Odlučio sam sve elemente kreirati u JavaScriptu.

Započeo sam kreiranjem HTML Canvasa u *setup()* funkciji. Njegovu širinu podesio sam na dimenzije visine prozora podijeljene s 1.414, a visinu na visinu prozora. Time sam dobio standardne proporcije papira A formata. Treći parametar *createCanvas()* funkcije je "SVG" koji određuje da se svi elementi Canvasa crtaju u vektorskom (SVG) formatu za razliku od standardnog moda crtanja koji je rasterski. Da bi koristio tu mogućnost morao sam u HTML uključiti biblioteku koja nadograđuje standardne funkcionalnosti p5.js-a, a naziva se *p5.svg.js* (https://github.com/zenozeng/p5.js-svg).

Sljedeći korak bio je zadati boju pozadine Canvasa funkcijom *background()*. Kod 10 označava skraćeni zapis HEX koda 10, 10, 10 koji je tamna nijansa sive.

U prijašnjoj sekciji 2. 3. 1. sam napomenuo kako se funkcija *draw()* izvršava 60 puta u sekundi. U mom programu funkcija *draw()* uopće nije potrebna pa sam ju isključio naredbom *noLoop()*.

Za dovršavanje izrade strukture web-aplikacije još je jedino preostalo izraditi HTML elemente iz desnog stupca (gumbe, slidere, …). Njih sam izradio u funkciji *createSideHtmlElements()* koju pozivam u *setup()* funkciji. Na većinu izrađenih HTML elemenata dodao sam i ime ID-a da bi ih poslije mogao referencirati u kodu te uređivati njihov dizajn pomoću CSS-a.

main.js

```
. . .
let bgColor;
. . .

function setup() {
  createCanvas(windowHeight/1.414, windowHeight, SVG);
  bgColor = color(10);
  background(bgColor);

  noLoop();
```

```
  createSideHtmlElements();
  . . .
}


. . .


function createSideHtmlElements() {
  createDiv().id('rightDivColumn');
  . . .
  createDiv().id('firstSectionDiv2').parent('rightDivColumn');
  createElement('h3', 'Shapes').parent('firstSectionDiv2');
  shapeGenerateButton = createButton('Generate').parent('firstSecti-
onDiv2');
  sliderNumberOfShapes = createSlider(1, 5, 3, 1);
  paragraphNumberOfShapes = createP('Number of shapes: ' + sliderNum-
berOfShapes.value()).parent('firstSectionDiv2');
  sliderNumberOfShapes.parent('firstSectionDiv2');
  . . .
}
```

U CSS datoteci podesio sam fontove, boje elemenata i pozadine cijele stranice te općenito dizajn i raspored elemenata.

Fontove sam uključio u aplikaciju preko servisa Google Fonts (https://fonts.google.com) naredbom *@import* te ih postavljao na elemente naredbom *font-family*.

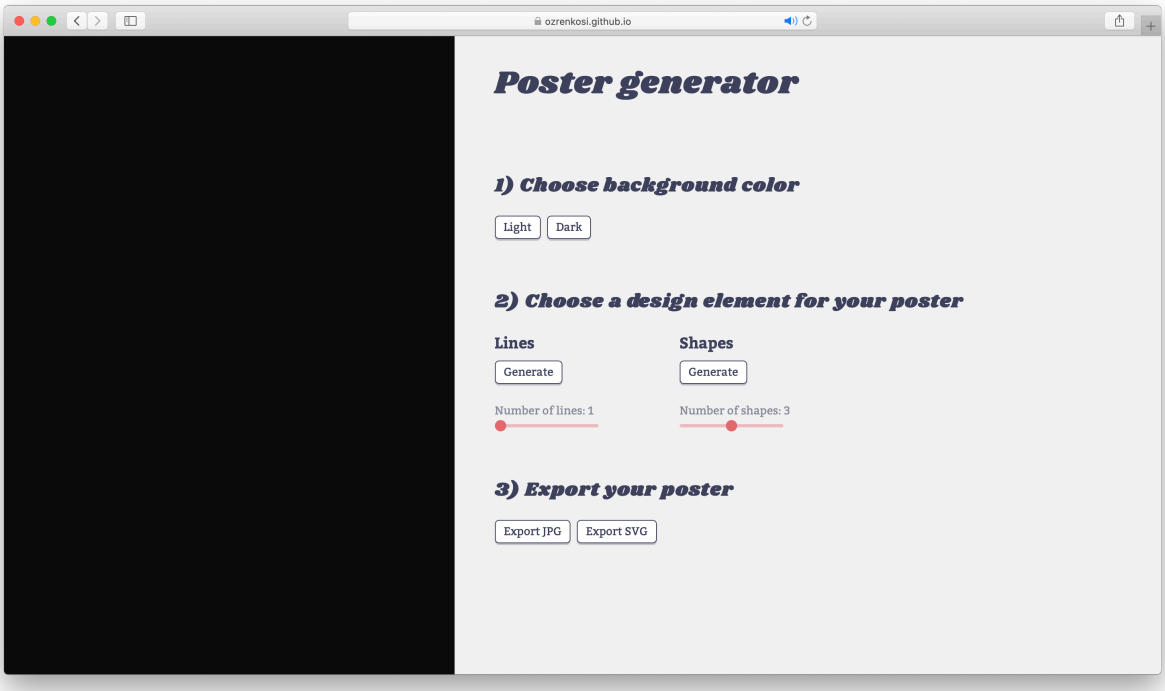
Dodatno bih istaknuo način korištenja boja u CSS-u. Koristio sam jednu od novih funkcionalnosti CSS-a, CSS Custom Properties. Pomoću njih mogu HEX kod boje spremiti u varijablu te zadavati boju elemenata preko nje. Prednost tog načina rada je ta da ako se naknadno odlučim promijeniti boju svim elementima tu promjenu moram učiniti samo jednom, a ne zasebno na svakom elementu.

style.css

```
@import url('https://fonts.googleapis.com/css2?family=Shrikhand&dis-
play=swap');
```

```
@import url('https://fonts.googleapis.com/css2?
family=Bitter:wght@400;700&display=swap');

/* Colors */
:root {
  --super-light-purple: #f4f1f4;
  --dark-blue: #3d405b;
  --red: #e56b6f;
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Bitter', serif;
  color: var(--dark-blue);
}
```

. . .



Slika 7 - Gotova vizualna struktura web-aplikacije

3.3. Izrada algoritama za crtanje postera

Ključni dio aplikacije je algoritam koji je zaslužan za crtanje dizajna postera. Napisao sam dva algoritma, jedan bazira dizajn na oblicima, drugi na linijama. Prvo ću objasniti algoritam koji crta oblike. Oba algoritma sam radio kao JavaScript objekte u zasebnim datotekama.

Koncept algoritma koji crta oblike je da na slučajnoj lokaciji nacrta ili pravokutnik ili elipsu slučajno odabrane boje i slučajnih dimenzija. Objekt za crtanje oblika sastoji se od dvije funkcije, konstruktor funkcije i *show()* funkcije. Konstruktor funkcija prilikom izrade objekta (oblika) postavlja njegovu boju na slučajno izabranu boju s blagom prozirnošću (alfa kanalom). s*how()* funkcija miče obrub oblika, postavlja njegovu ispunu na prije slučajno generiranu boju, postavlja početne koordinate crtanja pravokutnika na njegovo geometrijsko središte, postavlja koordinate cijelog Canvasa na njegovu sredinu, te s 50% šanse crta ili pravokutnik ili elipsu slučajnih dimenzija.

Shape.js

```
class Shape {
  constructor() {
    this.shapeColor = color(random(0, 255), random(0, 255), random(0,
255), random(150, 180));
  }

  show() {
    noStroke();
    fill(this.shapeColor);
    rectMode(CENTER);
    push();
    translate(width/2, height/2);

     if (random(1) > 0.5) {
      rect(random(-width/2, width/2), random(-height/2, height/2),
random(0.6*height, 0.9*height), random(0.06*height, 0.9*height));
    }
    else {
      ellipse(random(-width/2, width/2), random(-height/2, height/2),
random(0.25*height, 0.9*height));
    }
```

```
    pop();
    }
}
```

Drugi algoritam crtanja bazira se na generaciji linija čija je putanja određena Perlinovim šumom. Također se sastoji od konstruktor funkcije i *show()* funkcije. U konstruktor funkciji deklariram i inicijaliziram varijable koje će biti parametri algoritma. Tim načinom rada olakšavam čitanje algoritma te kontrolu parametara.

U *show()* funkciji za svaku liniju računam lokaciju svake točke linije prema nasumično generiranom Perlinovom šumu. Perlinov šum generiram pomoću funkcije *noise()*. Rezultat funkcije je broj između 0 i 1 koji skaliram na dimenzije Canvasa pomoću funkcije *map()*.

Line.js

```
class Line {
  constructor(lineWeight) {
    . . .
  }

  show() {
    . . .
    for (let i = 1; i <= this.numberOfSubLines; i++) {
      . . .
      beginShape();
      for (this.xPos = 0; this.xPos < width; this.xPos = this.xPos +
this.lineResolution) {
        this.yOff = map(noise(this.noiseOffset + this.xPos*this.no-
iseScrollSpeed), 0, 1, -this.lineOffsetMAX, this.lineOffsetMAX);
        if (this.yOff > 0) {
          vertex(this.xPos, this.yPos + this.yOff - map(i, 1,
this.numberOfSubLines, 0, abs(this.yOff)/2));
        }
        else {
          vertex(this.xPos, this.yPos + this.yOff + map(i, 1,
this.numberOfSubLines, 0, abs(this.yOff)/2));
        }
```

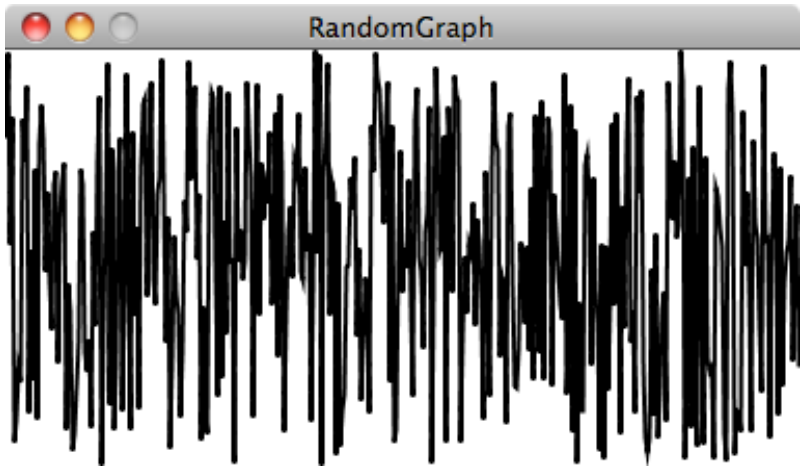
```
        }
        endShape();
      }
    }
  }
```

3.3.1. Perlinov šum

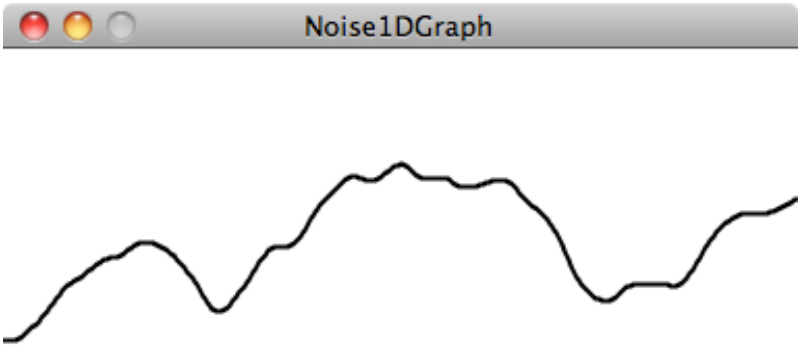
Dobar generator slučajnih brojeva proizvodi brojeve koji nemaju međusobni odnos i ne pokazuju nikakav uzorak. Mala slučajnost može biti dobra stvar kod programiranja organskih elemenata, međutim potpuna nasumičnost nije prirodna.

Postoji algoritam koji rezultira prirodnijim rezultatima, a poznat je pod nazivom Perlinov šum. **[15]** Ken Perlin razvio je algoritam dok je radio na filmu Tron u ranim 80-ima. Koristio ga je za izradu proceduralnih tekstura za računalno generirane efekte. Za razvoj algoritma Ken Perlin je 1997. godine osvojio priznanje za tehnička dostignuća. Perlinov šum može se koristiti za stvaranje elemenata s prirodnim kvalitetama, poput oblaka, pejzaža i tekstura s uzorkom poput mramora.

Za razliku od potpune nasumičnosti Perlinov šum izgleda više organski jer stvara prirodno uređen ("gladak") niz pseudo slučajnih brojeva.



Slika 8 - Graf potpune nasumičnosti kroz vrijeme



Slika 9 - Graf Perlinovog šuma kroz vrijeme

3.4. Povezivanje algoritama za crtanje postera s HTML elementima

Na HTML gumbe dodao sam funkciju koja na klik generira objekte (linije ili oblike, broj njih određen sliderom) te poziva funkciju *show()* za svaki od njih.

main.js

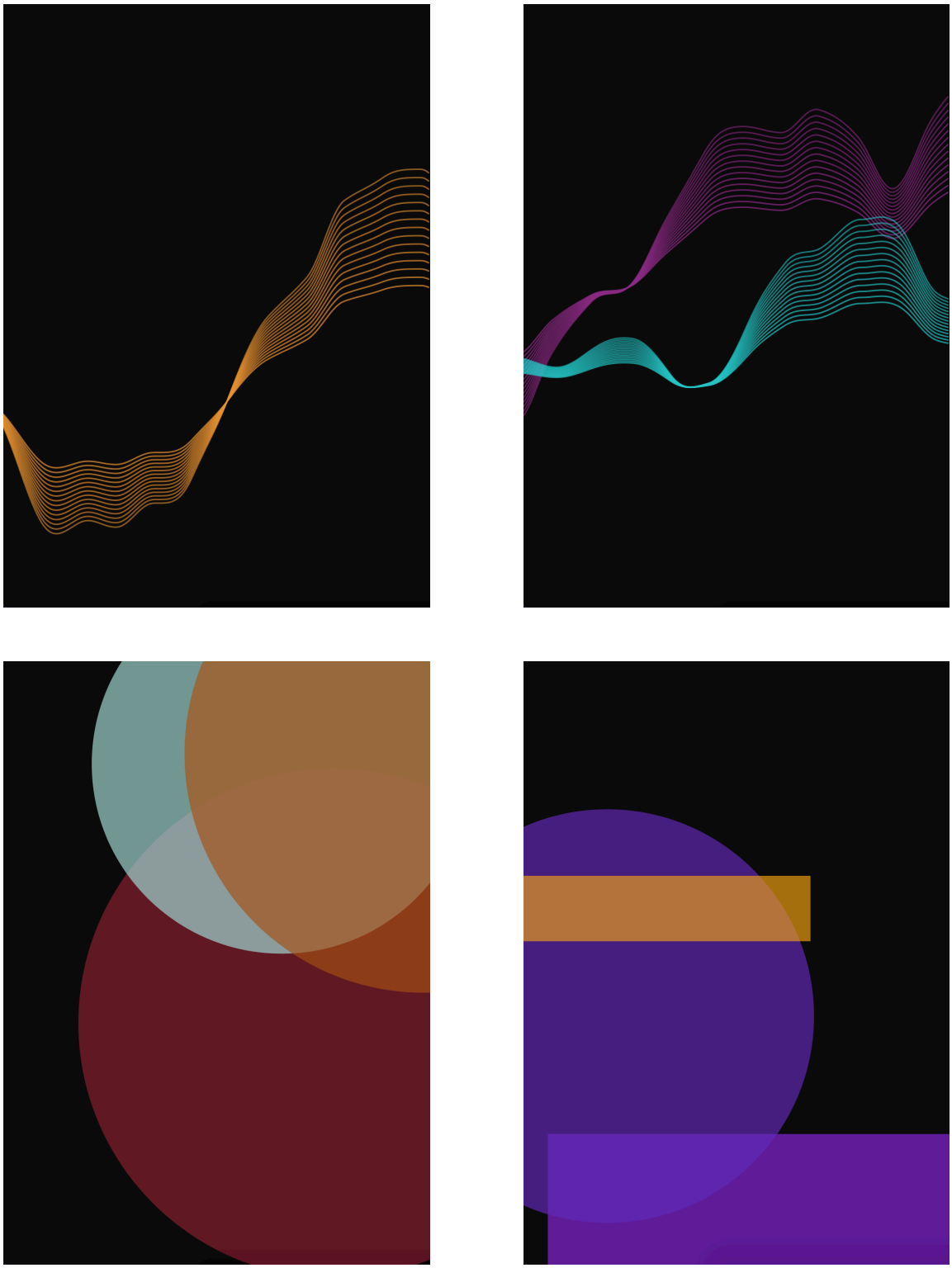
```
. . .
function setup() {
. . .
  shapeGenerateButton.mousePressed(function() {
    lines = [];
    shapes = [];
    background(bgColor);
    createShapes(sliderNumberOfShapes.value());
    drawShapes();
  });
. . .
}
. . .
function createShapes(numberOfShapes) {
  for (let i = 0; i < numberOfShapes; i++) {
    shapes[i] = new Shape();
  }
}
```

```
function drawShapes() {
  for (let i = 0; i < shapes.length; i++) {
    shapes[i].show();
  }
}
. . .
```

Time je programiranje web-aplikacije završeno i ona je spremna za korištenje. U idućem poglavlju ću prikazati dizajne postera generirane izrađenom aplikacijom.

4. Posteri generirani web-aplikacijom

U nastavku su prikazani dizajni generirani web-aplikacijom:



Slika 10 - Dizajni generirani web-aplikacijom

5. Zaključak

Zadatak ovog rada uspješno je izvršen. Korisnih web-aplikacije može generirati dizajn postera te ga čak pomoću par parametara uređivati prema vlastitoj želji. Korisnik ima opcije odabrati boju pozadine postera, oblik na kojem će se bazirati dizajn te broj pojavljivanja odabranog oblika. Kad generira dizajn s kojim je zadovoljan može ga eksportirati ili u rasterskom (JPG) ili u vektorskom (SVG) formatu te ga dalje uređivati u bilo kojem grafičkom editoru.

Web-aplikacija također može dizajneru pružiti izvor inspiracije, potaknuti daljnji razvoj generirane ideje. Aplikacija je postavljena na web i dostupna za korištenje na adresi http://arwen.unin.hr/~ozkosi/zavrsni-rad/index.html, a u budućnosti se može proširiti dodavanjem novih algoritama za crtanje.

Radeći na projektu stekao sam vrijedna znanja o izradi JavaScript/p5.js web aplikacija, rješavanju grešaka u kodu, povezivanjem koda s vanjskim bibliotekama, iteriranju algoritama, traženju potencijalnih rješenja problema na internetu, sve bitnim faktorima pri razvoju softwarea.

U Varaždinu, 28. rujna 2020.

6. Literatura

1. Tomšić, Damjan. HTML dokument i osnove HTML jezika. http://www.oblakznanja.com/2013/04/html-dokument-i-osnove-html-jezika/ (pristupljeno 12. srpnja 2020.).
2. Što je HTML?. Zagreb. https://tesla.carnet.hr/mod/book/view.php?id=5430&chapterid=885 (pristupljeno 12. srpnja 2020.).
3. Tim Berners-Lee. https://www.w3.org/People/Berners-Lee/ (pristupljeno 6. rujna 2020.).
4. Kyrnin, Jennifer. HTML5 Canvas Uses. New York. https://www.lifewire.com/why-use-html5-canvas-3467995 (pristupljeno 12. srpnja 2020.).
5. HTML Canvas Graphics. https://www.w3schools.com/html/html5_canvas.asp (pristupljeno 12. srpnja 2020.).
6. Mujadžević, Edin. Uvod u CSS. Zagreb. https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c220_polaznik.pdf (pristupljeno 12. srpnja 2020.).
7. CSS. https://hr.wikipedia.org/wiki/CSS (pristupljeno 12. srpnja 2020.).
8. Stančer, Denis. Osnove JavaScripta. Zagreb. https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c501_polaznik.pdf (pristupljeno 12. srpnja 2020.).
9. Zekić-Sušac, Marijana. JavaScript. Osijek. http://www.mathos.unios.hr/wp/wp2009-10/P8_Java.pdf (pristupljeno 12. srpnja 2020.).
10. Parrish, Allison. First steps with p5.js. https://creative-coding.decontextualize.com/first-steps/ (pristupljeno 13. srpnja 2020.).
11. Preis, Johannes. Introduction to p5.js. https://medium.com/comsystoreply/introduction-to-p5-js-9a7da09f20aa (pristupljeno 13. srpnja 2020.).
12. Processing Foundation. A Modern Prometheus. https://medium.com/processing-foundation/a-modern-prometheus-59aed94abe85 (pristupljeno 6. rujna 2020.).
13. Processing. https://processing.org (pristupljeno 6. rujna 2020.).
14. Processing (programming language). https://en.wikipedia.org/wiki/Processing_(programming_language) (pristupljeno 6. rujna 2020.).
15. Perlin noise. https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise (pristupljeno 19. srpnja 2020.).

7. Popis slika

IZJAVA O AUTORSTVU I SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi ~~tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu~~ pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, OZREN KOSI *(ime i prezime)* ~~pod punom moralnom~~, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/~~ica~~ završnog/~~diplomskog~~ (*obrisati nepotrebno*) rada pod naslovom WEB APLIKACIJA - GENERATOR POSTERA*(upisati naslov)* te da u ~~navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni~~ dijelovi tuđih radova.

Student/ica:
~~(upisati ime i prezime)~~

Ozren Kosi
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se ~~realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.~~

Ja, OZREN KOSI *(ime i prezime)* neopozivo izjavljujem da sam suglasan/~~na~~ s javnom objavom završnog/~~diplomskog~~ (*obrisati nepotrebno*) rada pod naslovom WEB APLIKACIJA - GENERATOR POSTERA *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Ozren Kosi
(vlastoručni potpis)