TEHNIČKO VELEUČILIŠTE U ZAGREBU

STRUČNI STUDIJ RAČUNARSTVA

Dominik Črnjak

Analiza alata p5.js za izradu generativne umjetnosti i web dizajna

ZAVRŠNI RAD 0246099070-10-2022-1

Zagreb, rujan, 2023.

TEHNIČKO VELEUČILIŠTE U ZAGREBU

STRUČNI STUDIJ RAČUNARSTVA

Dominik Črnjak

JMBAG: 0246099070

Analiza alata p5.js za izradu generativne umjetnosti i web dizajna

ZAVRŠNI RAD 0246099070-10-2022-1

Zagreb, rujan, 2023.

Sažetak

U ovom radu analizira se JavaScript biblioteka p5.js kreirana u svrhu povezivanja umjetničkog izražavanja i računalnog programiranja. Cilj je prikazati prednosti, mane i načine korištenja biblioteke p5.js u odnosu na druge programske alate namijenjene za kreativno programiranje. Programiranjem primjera vizualne generativne umjetnosti i vizualizacije podataka prikazati će se sintaksa p5.js biblioteke i sve njene najvažnije funkcionalnosti, kao i razina njene kompleksnosti te odgovoriti na temeljno pitanje je li p5.js biblioteka pravi alat za sve stvaratelje kojima je primarni cilj umjetničko izražavanje u modernom webu dizajnu. Biblioteka će se testirati i usporediti po arhitekturi, kompleksnosti programiranja, mogućnosti učenja JavaScript programskog jezika i implementacije na web stranicu.

Ključne riječi: generativna umjetnost, kreativno programiranje, web dizajn, JavaScript, p5.js biblioteka

Abstract

This paper analyzes the JavaScript library p5.js, created to bridge the gap between artistic expression and computer programming. It aims to showcase the advantages, disadvantages, and particular ways of using the p5.js library compared to other programming tools designed for creative coding. By programming examples of visual generative art and data visualization, this paper will demonstrate the syntax of the p5.js library and its key functionalities, as well as the level of complexity involved. It will address the fundamental question of whether p5.js is the right tool for all creators whose primary goal is artistic expression in modern web design. The library will be tested and compared based on parameters such as architecture, programming complexity, learning curve, and implementation on web pages.

Keywords: generative art, creative coding, web design, JavaScript, p5.js library

Sadržaj

Popis slika

Popis tablica

Popis isječaka programskog koda

1. Uvod

U današnjem suvremenom dobu, u kojem tehnologija svakim danom sve više napreduje, kao najvažnije vještine koje bi trebao posjedovati svaki čovjek, neovisno o zanimanju, sve se više ističu sposobnost kreativnog izražavanja te vještina analize i rješavanja problema[1]. Kreativno kodiranje je način programiranja kojemu je u fokusu kreativan pristup rješavanju problema i izražavanje ideja kroz sam programski kod[2], što upravo pridonosi razvitku spomenutih vještina. To podrazumijeva korištenje raznih tehnologija, programskih jezika i alata prilikom izrade projekata koji ne teže isključivo funkcionalnosti, već mogu iskazati i umjetničke vrijednosti.

U web okruženju, gdje se kao dominantan programski jezik za manipulaciju i prikaz web sadržaja uspostavio JavaScript, kao jedan od najkorišteniji alata za kreativno kodiranje, istaknula se JavaScript biblioteka p5.js. Zbog velike raznolikosti funkcija i naglaska na jednostavnost cilj je ispitati i analizirati p5.js biblioteku te utvrditi može li uspješno i kvalitetno obraditi zadatke različitih područja kreativnog kodiranja, a pritom ostati dovoljno razumljiva i strukturalno jednostavna da privuče početnike u programiranju i računalne umjetnike željne kreativnog izražavanja.

Praktični dio rada sastoji se od analize i usporedbe p5.js alata u dva najvažnija područja kreativnog kodiranja – izradi generativne umjetnosti te vizualizaciji podataka. Kako bi svi dijelovi izrade oba primjera bili analizirani u kontekstu modernog web okruženja p5.js će biti uspoređen s dvije relevantne JavaScript biblioteke koje se aktivno koriste u navedenim područjima – u izradi generativne umjetnosti i grafičkoj vizualizaciji s bibliotekom Paper.js te u području vizualizacije podataka s bibliotekom D3.js. Svi konačni rezultati, osobna mišljenja i komentari te tablica ocjena su objedinjeni u poglavlju rasprava rezultata.

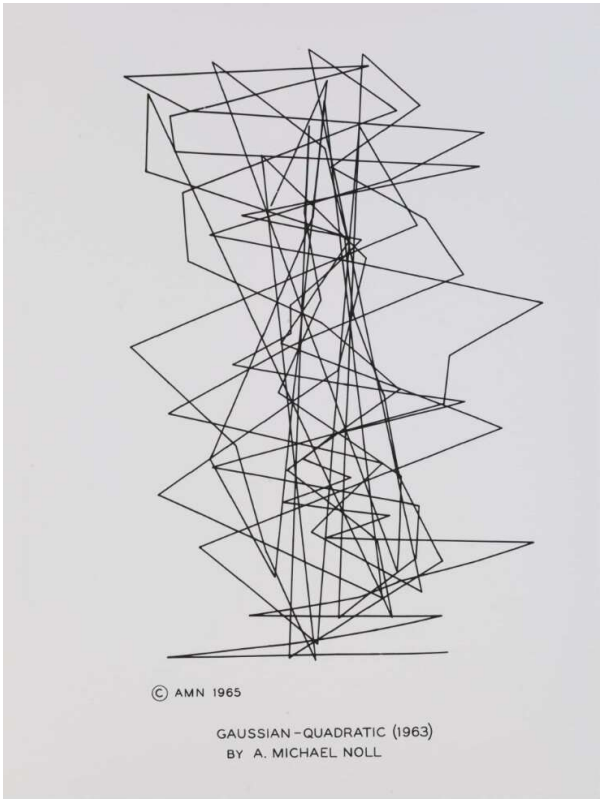
2. Kreativno kodiranje

U osnovi kreativno kodiranje je proces baziran na izražavanju, ponavljanju, refleksiji i otkrivanju novih postupaka u kojem se računalni kod koristi kao primarni medij za kreativno izražavanje[3]. Kreativno kodiranje se također može definirati kao spoj dvaju oprečnih pristupa programiranju i rješavanju problema, analitičkog i intuitivnog – u ovom slučaju kreativnog[4]. Analitički pristup je osnova svih znanosti pa tako i svih grana računalnog programiranja – proces koji se sastoji od analize problema, razvijanja rješenja na taj problem te njegove implementacije, dok je intuitivnost pristup koji nema strogo definiran način rješavanja problema, već se cijeli proces zbiva u hodu.

2.1. Povijest računalne umjetnosti

Još od ranih dana razvoja prvih računalnih uređaja bio je prisutan pojam “računalne umjetnosti” – što je u osnovi pojam vrlo blizak kreativnom kodiranju, gdje se uporabom novonastale tehnologije računalnog stroja nastojalo izraziti određene umjetničke ideje i procese. Početkom 1960ih godina započinju intenzivnija istraživanja inženjera i umjetnika koji nastoje pronaći načine kako računalni stroj upotrijebiti kao još jedan od alata pomoću kojih bi mogli prenijeti svoje umjetničke tendencije u tehnološki svijet.

Tako je 1962. godine američki inženjer A. Michael Noll programirao digitalno računalo koje generira različite vizualne uzorke bazirane na matematičkim algoritmima, što ga čini jednim od prvih računalnih uređaja kojemu je temeljna zadaća umjetničko stvaralaštvo[5]. Jedno od njegovih najpoznatijih djela pod nazivom “Gaussian-Quadratic” (slika 1.) bazirano je na algoritmu koji koordinate crteža duž vodoravne osi izračunava pomoću pseudo-slučajne Gaussove subrutine, dok se koordinate duž okomite osi računaju pomoću kvadratne jednadžbe.



Slika 1. *A. Michael Noll, "Gaussian-Quadratic"*

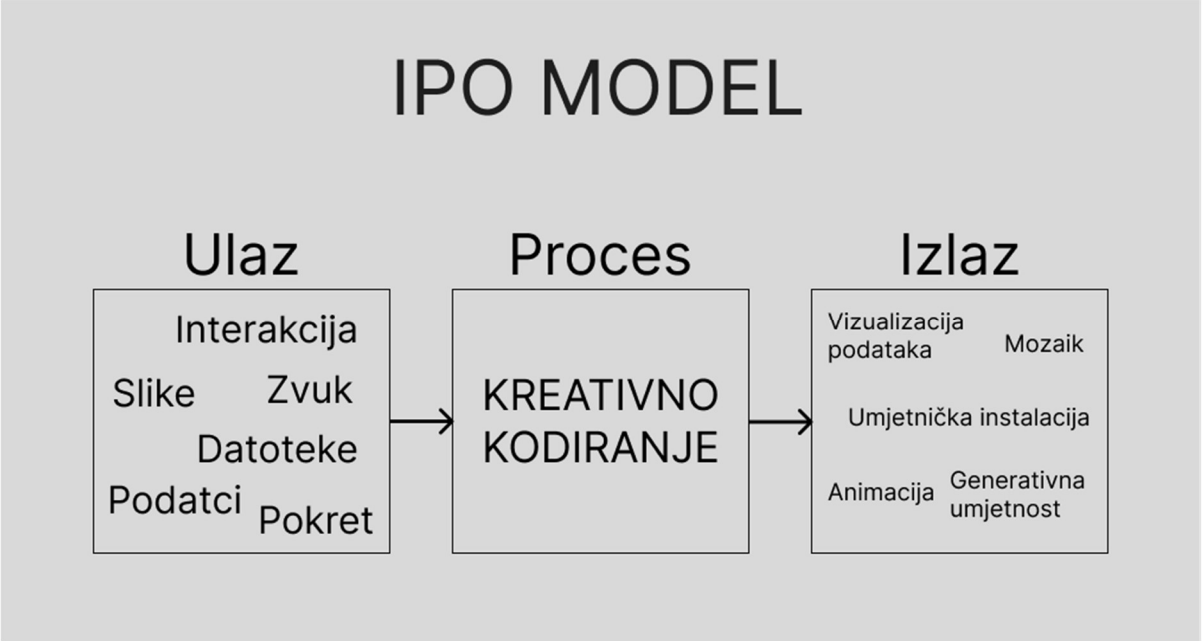
Jedan od prvih programskih jezika za kreativno kodiranje bio je Logo – programski jezik koji je u svojim začecima korišten za kontrolu robota koji stvara linijske crteže na papiru, kasnije prenamijenjen u alat za crtanje računalnih grafika, što mu je uvelike povećalo popularnost tijekom 1980ih godina te je postao objektom brojnih istraživanja na MIT-u (Massachusetts Institute of Technology)[6].

Kasnije, tijekom 1990ih godina također u američkom MIT-u, održan je eksperimentalni projekt pod nazivom "Design and numbers" tijekom kojeg su studenti MIT-a razvili software za učenje programiranja, koristeći principe kreativnog kodiranja koji je bio namijenjen svim umjetnicima, dizajnerima i početnicima u programiranju. Taj projekt imao je velik utjecaj na sve buduće programske jezike i projekte kreativnog kodiranja te je značajno utjecao na tvorce programskog jezika Processing-a koji je osnova svih važnijih modernih alata za kreativno kodiranje[7].

2.2. Primjena kreativnog kodiranja u obrazovanju

Kreativno kodiranje, osim samog umjetničkog izražavanja, također ima širu društvenu ulogu, jer može pomoći u razvoju tehnološke pismenosti i promovirati STEM (znanost, tehnologija, inženjerstvo i matematika) područja među mlađom populacijom, koja često preferira vizualne metode učenja s fokusom na kreativnost[8].

U osnovi, koncept kreativnog kodiranja je baziran na modelu “ulaz-proces-izlaz”(engl. IPO-model), gdje od određenih ulaznih podataka i/ili parametara (npr. teksta, brojeva, korisničke interakcije, matematičkih funkcija itd.) implementacijom određenog programskog rješenja(procesa) dobivamo željeni rezultat(izlaz)[9]. Navedeni model je glavni temelj za daljnje razumijevanje svih procesa u softverskom inženjerstvu, a njegova primjena u kreativnom kodiranju je vrlo intuitivna te je poveznicu ulaznih parametara i rezultata lako vizualno uočiti i razumjeti. Upravo zbog toga se kreativno kodiranje smatra dobrom “odskočnom daskom” za daljnje napredovanje u programiranju i učenju programske sintakse[10].



Slika 2. *Primjer IPO modela u kreativnom kodiranju*

2.3. Generativna umjetnost

Jedna od najznačajnijih primjena kreativnog kodiranja je izrada generativne umjetnosti tj. stvaranje vizualnih ili auditivnih djela korištenjem algoritama i parametara, zadanih od strane programera/umjetnika[11]. Iako se često u istom kontekstu spominju pojmovi generativne umjetnosti i AI umjetnosti, vrlo je važno razlikovati ta dva pojma. Generativna umjetnost se fokusira na korištenje algoritama i određenih pravila koje zadaje programer i time u potpunosti kontrolira postupak prema kojem će se djelo generirati, dok je AI umjetnost bazirana na algoritmu umjetne inteligencije koji je treniran na velikoj količini podataka (postojećim umjetničkim slikama, fotografijama i sl.) prema kojima zadaje parametre i pravila generacije[12].

U biti, princip generativne umjetnosti omogućuje umjetniku/programeru da se fokusira na kreativan proces i pronalazi inovativne načine na koje se može izraziti korištenjem programskog koda i algoritama, neovisno o krajnjem rezultatu. Generativna umjetnost otvara nove mogućnosti svim umjetnicima koji su spremni prihvatiti tehnologiju, kao još jedan od alata za stvaranje umjetnosti, ali i proširuje ulogu programskog koda kao medija koji potiče kreativnost i može biti sastavni dio kreativnog procesa[13].

Još jedan od načina, na koji kreativno kodiranje može pomoći u umjetničkom izražavanju, jest implementacija interaktivnosti u već postojeća ili novonastala umjetnička djela, što može dovesti do novih oblika umjetnosti i uspostaviti mogućnost interakcije umjetnika s publikom[14]. Temeljni postupak pri izradi interaktivnih instalacija je proces mapiranja – povezivanja i preslikavanja vrijednosti iz jedne domene u drugu, što omogućuje obradu podataka iz okoline (npr. glasovi, dodiri, kretnje) te njihovu pretvorbu u nove audio-vizualne oblike[15].

2.4. Kreativni proces vizualizacije podataka

Iako je česta primjena kreativnog kodiranja u umjetničke svrhe, kreativno se može izraziti te pritom ostvariti rezultat koji može biti iskorišten u praktične svrhe. Vizualizacija podataka je proces pretvaranja korisniku apstraktnih podataka, brojčanih i statističkih, u vizualno atraktivne grafičke prikaze koji pomažu u razumijevanju i interpretaciji informacija[16]. Podaci su osnovni elementi koji se koriste u analitičkim procesima i donošenju poslovnih odluka, a oni mogu biti u obliku tekstualnih datoteka, tablica, baza podataka ili drugih oblika strukturiranih podataka.

Važna uloga kreativnog kodiranja u vizualizaciji podataka je implementacijom koda prikazati podatke na inovativne načine. Za razliku od tradicionalnih načina vizualizacije podataka (grafovima, tablicama i sl.) kreativno kodiranje nudi mogućnosti vizualizacije podataka na umjetnički ekspresivnije načine[17], koji mogu pridonijeti izražavanju neke dublje ideje ili utjecati na kritičko mišljenje kod publike[18]. Osim toga, kreativno kodiranje ima prednost u svojoj generativnosti, odnosno sposobnosti računala da obrađuje tisuće iteracija koda u sekundi i pritom obradi veliku količinu podataka. Kvalitetnim i inovativnim procesima vizualizacije podataka možemo pomoći u izražavanju ideja, kreiranju važnih poslovnih odluka, čak i izražavanju političkih stavova[19].

2.5. Alati za kreativno kodiranje

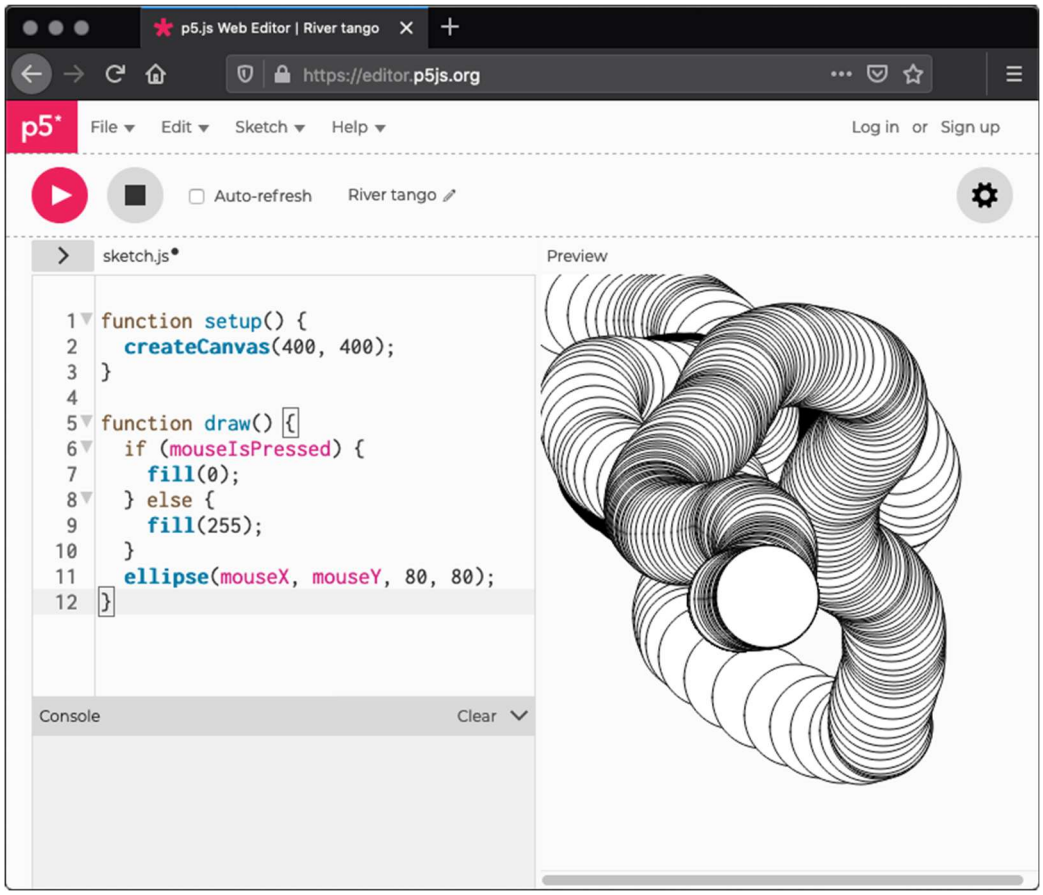
Postoji velik broj alata, programskih jezika i biblioteka, koji su razvijeni isključivo u svrhe kreativnog kodiranja. Dok su neki od alata prilagođeni specifičnim namjenama – poput Orce[20], alata namijenjenog za izradu generativne glazbe, većina je višenamjenska i sadrži mnoge funkcije koje pojednostavljuju razne kreativne postupke(poput vizualizacije, interaktivnosti, animacije). Među takvim višenamjenskim alatima za kreativno kodiranje ističu se alati iz obitelji Processing – posebice Processing programsko okruženje te njegova web inačica u vidu JavaScript biblioteke p5.js.

2.5.1 Programski jezik Processing

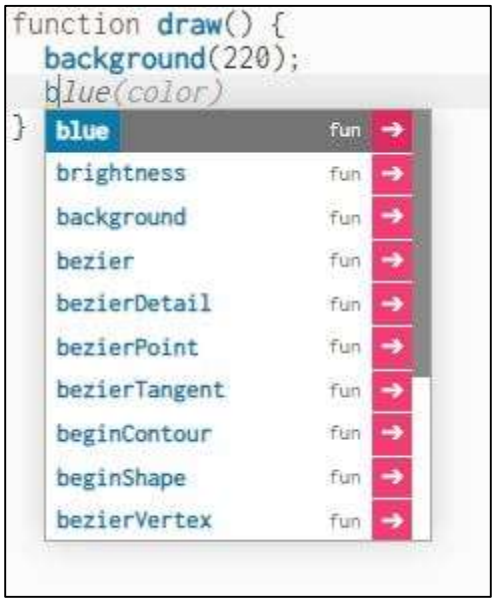
Processing[21] – programski jezik i razvojno okruženje bazirano na objektno-orijentiranom programskom jeziku Java, jedan je od prvih i najpoznatijih višenamjenskih alata kojima je glavna namjena i fokus kreativno kodiranje[3]. Mogućnosti Processinga su višestruke zahvaljujući širokoj i aktivnoj zajednici koja je pridonijela stvaranju mnogobrojnih biblioteka i alata koji proširuju primjenu korištenja izvan osnovnog spektra – sposobnost obrade audio zapisa, mogućnost izrade kompleksnih animacija i grafika te vizualizacija velikih skupova podataka. Tijekom godina razvoja, Processing zajednica je razvila sučelja za programiranje u drugim programskim jezicima kao što su Python (Processing.py)[22], Ruby (Ruby Processing)[23] te JavaScript (p5.js)[24].

2.5.2 JavaScript biblioteka p5.js

Biblioteka p5.js programskog okruženja JavaScript, jedna je od najkorištenijih i najdokumentiranijih alata za kreativno kodiranje u web okruženju. Prema istraživanju[25] provedenom među programerima koji su imali iskustva s p5.js bibliotekom, kao glavni aduti se ističu jednostavnost korištenja, lakoća učenja i pristupačnost, dok kao glavne nedostatke navode lošije performanse i manjak naprednijih značajki. Prema ispitanicima p5.js biblioteka najviše je bila korištena u područjima umjetničkog izražavanja, web-dizajna te vizualizacije podataka, a kao neke od alternativa u tim područjima istaknuli su biblioteke Paper.js i D3.js. Zbog svoje jednostavnosti i lakoće učenja p5.js biblioteka je često korištena u raznim edukacijskim projektima[8], eksperimentalnim programima te uvrštena u mnoge školske kurikulume, a svojim web-orijentiranim programskim okruženjem(slika 3.) koje nudi instant vizualnu povratnu informaciju i brojne funkcije koje olakšavaju korištenje, poput funkcije automatskog upotpunjavanja naredbi (slika 4.), djeluje motivirajuće na sve početnike u programiranju[26].



Slika 3. *p5.js web sučelje*



Slika 4. *Funkcija za automatsko upotpunjavanje naredbi*

2.5.3 JavaScript biblioteka Paper.js

Paper.js je JavaScript biblioteka za izradu i manipulaciju vektorskim grafikama, čiji se kod izvodi na HTML5 Canvas elementu. Pruža čistu strukturu programskog koda korištenjem vlastite ekstenzije JavaScript jezika – PaperScripta, moćne funkcionalnosti za stvaranje i manipulaciju vektorskih grafika te sadrži širok broj metoda koje olakšavaju procese kreativnog kodiranja.

Zahvaljujući otvorenosti koda i aktivnoj zajednici, biblioteka se redovito ažurira i unapređuje, no kako je Paper.js relativno nova biblioteka o njoj nema velik broj radova i usporedba s drugim alatima za kreativno kodiranje i izradu vektorskih grafika.

2.5.4 JavaScript biblioteka D3.js

Biblioteka D3.js[27] (engl. Data-Driven Documents) jedna je od najkorištenijih alata u području vizualizacije i obrade podataka u web okruženju. Pomoću D3.js-a, ulazni podatci mogu biti povezani s DOM (Document Object Model) elementima, omogućujući manipulaciju podacima i izvođenje transformacija na samom HTML dokumentu pritom podržavajući širok spektar različitih formata podataka, uključujući CSV, JSON i GeoJSON.

Za razliku od p5.js biblioteke, D3.js nije namijenjen za potpune početnike u programiranju te za stvaranje grafika i vizualizacija zahtijeva osnovno poznavanje HTML-a, JavaScripta, CSS-a i SVG-a. Čak i za stvaranje vrlo jednostavnih vizualizacija potrebno je isprogramirati funkcije niže razine (engl. low level), što može uvelike produžiti vrijeme razvoja, budući da D3.js nema unaprijed ugrađenu biblioteku grafova ili dijagrama za odabir. Međutim, D3.js dolazi s velikim brojem ugrađenim funkcija za matematičku manipulaciju podataka i dokumentacijom bogatom primjerima koji se mogu ponovno koristiti te značajno olakšati vizualizaciju podataka. Kako se u D3.js cijeli proces vizualizacije bazira na prikazu vektorskih elemenata i korištenjem SVG formata, poznavanje vektorske grafike je ključan korak u savladavanju korištenja D3.js kao konkurentnog alata[28].

3. Metodologija

Biblioteka p5.js će se ispitati i analizirati programiranjem dvaju praktičnih primjera - programa za izradu generativne umjetnosti prema slikovnom predlošku te programa koji vizualizira područja, točke interesa i njihove relacije na geografskoj karti. Kako je p5.js biblioteka sa širokim opsegom funkcionalnosti, fokus analize će biti isključivo na funkcionalnostima koje se koriste u praktičnim primjerima.

Kako bi se pobliže analizirale funkcionalnosti, osnovna sintaksa i kompleksnost p5.js će se usporediti s JavaScript bibliotekama koje se u području izrade generativne umjetnosti i vizualizacije podataka često navodi kao glavne alternative – Paper.js i D3.js. Također, vrlo važna stavka svakog programskog alata i biblioteke, koja će se kodiranjem primjera analizirati, jest sposobnost učenja te intuitivnost programske sintakse, koja je ovdje vrlo važna stavka, s obzirom da je imperativ većine alata za kreativno kodiranje omogućiti kreativno izražavanje, neovisno o prethodnom programerskom iskustvu.

Kako bi konačni rezultati analize i usporedbe bili prikazani na razumljiv način za svaki od parametara, prema kojima se provodi analiza, bit će izražena subjektivna ocjena prema brojčanoj skali od 1-5, oviseći o sveukupnom dojmu, funkcionalnosti i broju mogućnosti koje određena biblioteka nudi. Ako, primjerice, biblioteka u području neke od analiziranih funkcionalnosti sadrži samo osnovne JavaScript metode i/ili ne olakšava izradu praktičnog primjera, njena ocjena za taj parametar ne može biti veća od ocjene 1. U suprotnome, ako biblioteka obiluje dodatnim funkcijama i mogućnostima te uvelike olakšava izradu za taj parametar, može ostvariti maksimalnu ocjenu 5. Ocjene će biti prikazane u konačnoj tablici rezultata te će se ukupnom srednjom vrijednosti utvrditi koji se alat pokazao boljom opcijom prilikom programiranja primjera generativne umjetnosti i vizualizacije podataka.

3.1. Primjer generativne umjetnosti

U primjeru izrade generativne umjetnosti p5.js alat će se testirati te usporediti s bibliotekom Paper.js. Izradom generatora generativne umjetnosti od učitane slike ispitat će se osnovna arhitektura i sintaksa p5.js i Paper.js biblioteka te njihove mogućnosti u obradi slika, manipulaciji DOM elemenata i korištenja funkcija nasumičnosti. Također, analizirat će se različiti pristupi prikazu elemenata te usporediti koji je pristup (vektorski/rasterski) optimalniji za izradu generativne umjetnosti i obradu slika.

3.1.1. Arhitektura i kontekst izvršavanja

Biblioteke p5.js i Paper.js se interpretiraju u JavaScript skriptnom jeziku te za prikaz programskog koda koriste HTML5 Canvas (platno) element. Iako web-preglednik obje biblioteke interpretira kao JavaScript, Paper.js je izvorno napisan u Paperscriptu, ekstenziji JavaScripta koja omogućuje rad na višoj razini apstrakcije. PaperScript uvodi poseban objekt nazvan PaperScope, koji služi kao zasebni izvršni kontekst (engl. execution context) za izvršavanje svih Paper.js skripti. Svaki PaperScope objekt može sadržavati jedan ili više projekata, koji se sastoje od platna i elemenata koji se renderiraju na tom platnu, također sadržavajući vlastiti set alata koji se koriste za manipuliranje elementima na projektu, kao i druge specifične postavke i funkcije. Korištenjem PaperScripta i PaperScope objekata, programer može lako izolirati svaki projekt i njegov kontekst izvršavanja od drugih projekata koji se mogu nalaziti na istoj stranici, smanjujući šanse za greške i konflikte koje se mogu pojaviti kada se više projekata pokuša manipulirati istim skupom funkcija.

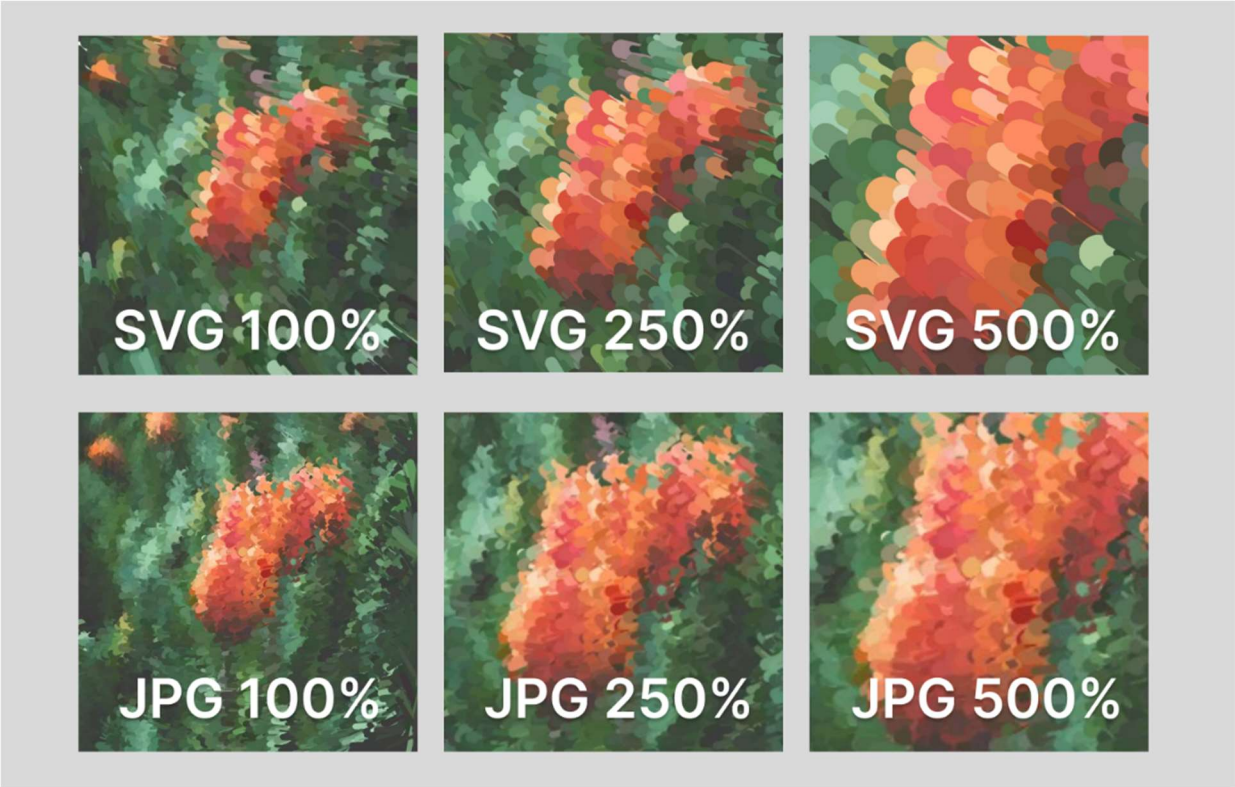
Usporedno s time, u čistom JavaScriptu kojeg koristi p5.js biblioteka, programski kod se izvršava u globalnom kontekstu izvršavanja. To znači da će sve varijable, funkcije i objekti koji su definirani u programu biti vidljivi u globalnom opsegu, što može dovesti do konflikata i grešaka pri radu na više projekata istovremeno. Zbog toga je preporučljivo koristiti “instance mode” pomoću kojega p5.js omogućava stvaranje izoliranih instanci projekata.

3.1.2. Analiza rasterskog i vektorskog prikaza grafičkih elemenata

Biblioteka p5.js sve elemente na platnu prikazuje rasterski – svaki je element prikazan pomoću mreže piksela te je rezolucija fiksna, dok Paper.js u fokusu ima vektorski pristup prikazu i manipulaciji grafikom. Vektorski pristup se temelji na objektima poput točaka, linija, krivulja i oblika koji se definiraju putem matematičkih formula koje određuju njihov položaj, oblik, veličinu i stilizaciju. Kako bi se ti elementi prikazali uistinu vektorski, potrebno je koristiti ugrađene Paper.js metode koje pretvaraju grafički prikaz u SVG format.

Najveća prednost korištenja vektorskog prikaza u web okruženju je u tome što elementi grafike, zbog toga što su matematički definirani, ostaju oštri bez obzira na skaliranje - grafika se može se povećavati ili smanjivati bez gubitka kvalitete. To čini vektorski prikaz prikladnim za prikaz elemenata koji trebaju biti prilagodljivi na razne veličine zaslona i uređaja, primjerice logotipa. Isto tako, vektorske objekte moguće je zasebno stilski oblikovati i manipulirati, izravno u SVG tekstualnoj datoteci ili pomoću vanjskih CSS datoteka, što pridonosi velikoj fleksibilnosti stiliziranja svake vektorske grafike i njenih elemenata.

S druge strane, rasterski prikaz pruža veću fleksibilnost u obradi i manipulaciji pojedinačnim pikselima slike te mogućnost korištenja različitih efekata poput zamućenja, promjene boja, primjene filtera i drugih tehnika obrade slika. Rasterski formati, kao što su JPEG, PNG ili GIF, su široko podržani na različitim web preglednicima i uređajima, što osigurava da će slike biti prikazane ispravno na većini platformi bez problema s kompatibilnošću. Za razliku od vektorskog prikaza rasterski formati su idealni za prikaz fotografija jer zadržavaju visoku kvalitetu detalja, boja i prijelaza te su općenito prvi izbor za prikaz većine fotografija na webu.



Slika 5. *Usporedba vektorskog i rasterskog formata pri različitim uvećanjima*

Format	SVG	JPG
Mogućnost stilizacije	5	3
Razlučivost	5	3
Jednostavnost implementacije	3	5
Prikaz detalja	3	4
Rad sa slikama	3	5
Ukupna ocjena	**3,8**	**4,0**

Tablica 1. *Rezultati korištenja vektorskog i rasterskog formata na primjeru*

3.1.3. Opća sintaksa za izradu i manipulaciju grafičkim elementima

Sintaksa p5.js biblioteke je vrlo jednostavna i intuitivna, gdje je glavni koncept svakog programa podijeljen u dvije funkcije – draw() i setup(). Setup() funkcija se izvršava samo na početku programa te služi za inicijalizaciju početnih postavki projekta, njegovo pozicioniranje na ekranu te inicijalizaciju svih potrebnih HTML elemenata. Draw() funkcija se izvršava kontinuirano, izvodeći se u jednoj neprekinutoj petlji te služi za iscrtavanje elemenata na platno i podržava mogućnost animacije.

Iako se p5.js može koristiti za stvaranje objekata i služiti se objektno-orijentiranim konceptima, poput definiranja klasa i stvaranja instanci objekata, sam p5.js pristupa programiranju na način koji olakšava manipulaciju vizualnim elementima putem funkcija. Tako primjerice za iscrtavanja linija u primjeru dovoljno je koristiti ugrađenu metodu curve() koja za parametre prima koordinate kroz koje će se krivulja kretati.

S druge strane, Paper.js koristi objektno-orijentiran pristup manipulaciji grafičkim elementima. Najprije je potrebno stvoriti projekt u obliku objekta te ga inicijalizirati na odgovarajuće platno koristeći funkciju Paper.setup(), nakon čega je na Paper objektu moguće stvoriti objekte kao što su Layer(sloj), Path(putanja), Point(točka) i Raster za manipulaciju grafičkim elementima.

Za programiranje primjera koristit će se osnovni objekt Path(putanja) koji se koristi za definiranje i manipulaciju vektorskih putanja. Putanja se sastoji od niza točaka (kontrolne točke i točke puta) i segmenata koji se koriste za stvaranje krivulja, linija i ostalih geometrijskih oblika. Putanja može biti otvorena ili zatvorena (prva i posljednja točka su povezane).

```
function setup() {
        canvas = createCanvas(width, height);
}
function draw() {
        curve(x1, y1, x2, y2, x3, y3);
}
```

Isječak programskog koda 1. *Sintaksa za izradu jednostavne krivulje – p5.js*

```
var path = new paper.Path();
var startPoint = new paper.Point(x,y);
path.moveTo(startPoint);

var handlePoint1 = new paper.Point(x, y);
var handlePoint2 = new paper.Point(x, y);
var endPoint = new paper.Point(x, y);

var curveSegment = new paper.Segment(handlePoint1, handlePoint2, endPoint);
path.add(curveSegment);
```

Isječak programskog koda 2. *Sintaksa za izradu krivulje – Paper.js*

3.1.4. Manipulacija i rad sa slikovnim podatcima

Kako bi obrada slika u p5.js-u bila moguća najprije ih je potrebno učitati, što je moguće učiniti koristeći funkciju loadImage() koja omogućava učitavanje slika iz različitih izvora poput lokalnog računala ili određene web adrese. Nakon što je slika učitana, moguće ju je prikazati na platnu pomoću funkcije image() te funkcijama get() i set() pristupiti pikselima slike i dohvaćati / manipulirati njihovim vrijednostima.

Za potrebe primjera, pomoću funkcije get() moguće je dohvatiti boju piksela u određenoj točki, koja je izražena u RGB vrijednosti te ju zatim iskoristiti kao boju zakrivljene linije koja će biti iscrtana na tom dijelu platna i reprezentirati određeni dio slike. Također p5.js nudi ugrađene funkcije za obradu slika poput tint(), filter() i blend(), koje manipulacijom određenih piksela na slici utječu na promjenu boja, primjenu filtera te određivanje vrste miješanja boja.

Obrada slike u Paper.js zahtjeva transformaciju učitane slike iz rasterskog oblika u vektorski stvaranjem Raster objekta kako bi sva daljnja obrada bila moguća. Pristupanje boji piksela na određenoj koordinati (x, y) je moguće pomoću metode getPixel() koja za razliku od p5.js biblioteke ne vraća RGB vrijednost piksela već objekt Color iz kojeg je naknadno moguće dobiti vrijednosti boje. Postoje biblioteke i alati, poput Potrace i ImageTrace.js, koji omogućavaju pretvaranje raster slika u vektorske oblike, koje se mogu dalje manipulirati korištenjem Paper.js-a te olakšati prethodno spomenutu transformaciju.

```
function setup() {
  input = createFileInput(handleFile);
}
function handleFile(file) {
   img = loadImage(file.data)
}
```

Isječak programskog koda 3. *Učitavanje slike – p5.js*

```
var file = document.getElementById('input').files[0];
var reader = new FileReader();
reader.readAsDataURL(file);

reader.onload = function (event) {
var img = new Image();
img.src = event.target.result;

var raster = new paper.Raster(img);
}
```

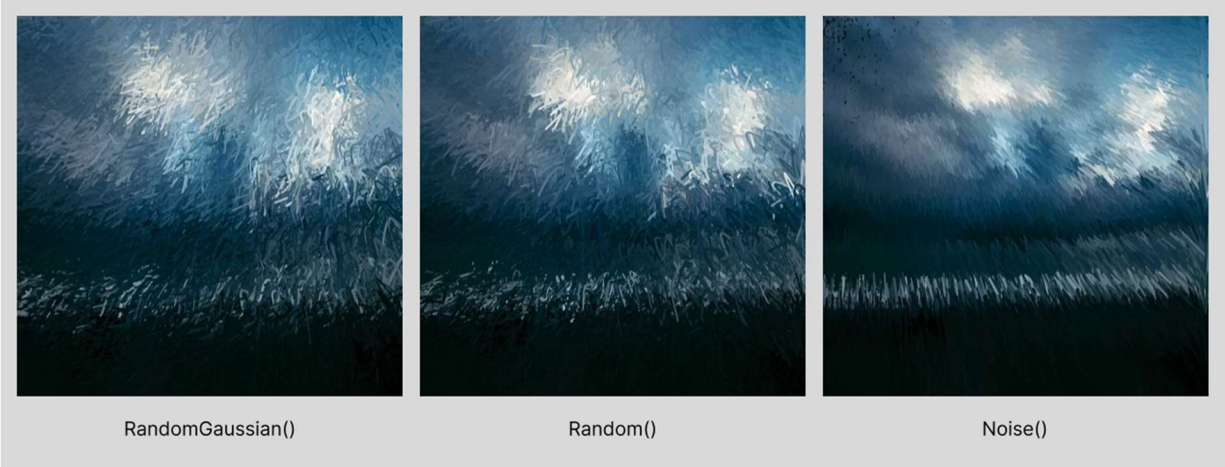
Isječak programskog koda 4. *Učitavanje slike – Paper.js*

3.1.5. Matematičke funkcije nasumičnosti

Pri izradi generativne umjetnosti, jedan od najvažnijih parametara jest upotreba funkcija za nasumičnost koje svako novo-generirano djelo čine jedinstvenim. U p5.js biblioteci glavna funkcija za generiranje slučajnih brojeva je random(), koja vraća decimalni broj između 0 i 1, kojoj je moguće zadati i vlastite granice generacije pomoću ulaznih parametara.

Također postoje i dodatne funkcije koje se mogu koristiti za generiranje slučajnih brojeva, kao što su noise(), koja koristi algoritam Perlinovog šuma za generaciju brojeva fluidnijih i prirodnijih razlika, te randomGaussian() koja generira nasumične brojeve prema Gaussovoj (normalnoj) distribuciji, koji gravitiraju prema zadanoj srednjoj vrijednosti.

Paper.js ne nudi dodatne funkcije za generiranje slučajnih brojeva, osim standardnih JavaScript funkcija za matematičke operacije – Math.random().



Slika 6. *Različite funkcije nasumičnosti u p5.js*

3.1.6. Manipulacija i rukovanje DOM elementima

DOM (engl. Document Object Model) je programsko sučelje koje predstavlja sve elemente na web stranici, uključujući osnovne HTML elemente kao što su naslovi, paragrafi, slike, forme, linkovi, kao i JavaScript objekte i događaje. DOM sučelje u osnovi omogućava da se vanjske skripte i programi, poput JavaScript-a, mogu povezivati i manipulirati elementima web stranice.

Biblioteka p5.js omogućava stvaranje, modifikaciju i stilizaciju HTML elemenata izravno u kodu, uključujući kontrolu ulaznih HTML elemenata kao što su tekstualna polja, gumbi i izbornici. p5.js također olakšava interakciju s DOM elementima, omogućavajući postavljanje događaja(engl. event) na ulaznim elementima i pristupanje vrijednostima unesenim u ulazne elemente. Iako je uobičajena praksa da se svi HTML elementi stvaraju i strukturiraju unutar same HTML datoteke, stvaranje ulaznih elemenata pomoću p5.js biblioteke pojednostavnjuje proces testiranja novo-nastalog programa te djeluje intuitivno svim početnicima u programiranju koji još nisu sasvim upoznati sa strukturiranjem i manipulacijom HTML elemenata.

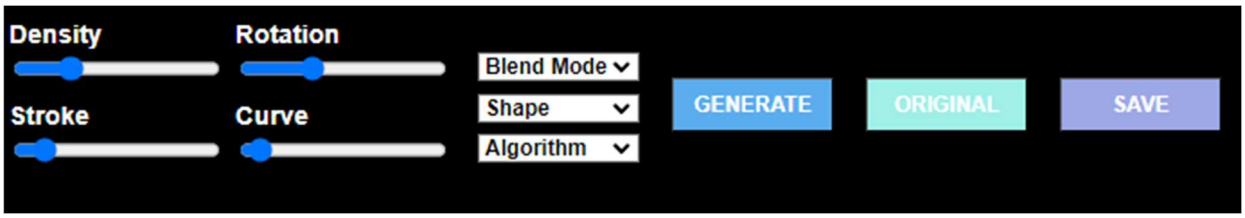
Paper.js nudi ograničenu podršku za manipulaciju DOM elementima, omogućujući postavljanje i promjenu stilova HTML elemenata, ali nije namijenjen stvaranju složenih DOM struktura ili interakciji s njima. Za stvaranje elemenata poput HTML ulaznog elementa (engl. input) u kodu, Paper.js se oslanja na funkcionalnosti DOM-a dostupne u JavaScriptu ili drugim bibliotekama.

```
let strokeLabel = createElement("Stroke", "Stroke");

strokeLabel.position(10, 60);

strokeSlider = createSlider(1, 20, 3, 1);

strokeSlider.position(10, 80);



value = strokeSlider.value()  //dohvaćanje ulazne vrijednosti
```

Isječak programskog koda 5. *Stvaranje „Slider“ HTML elementa - p5.js*

3.1.7. Realizacija praktičnog primjera generativne umjetnosti

Nakon što je slika učitana i podatci o bojama njenih piksela dohvaćeni, upotrebom funkcija za nasumičnost i funkcija za iscrtavanje krivulje moguće je realizirati praktični primjer izrade generativne umjetnosti. Također unutar p5.js programa su instancirani potrebni HTML ulazni elementi koji će služiti za određivanje parametara same generacije.



Slika 7. *Ulazni elementi potrebni za generaciju*

Kada je provjera učitavanja slike obavljena (kondicijska varijabla koja predstavlja sliku je postavljena na „true“) moguće je iteracijom pomoću petlje, koja ovisi o unesenoj gustoći piksela (Density slider element), dohvatiti podatke slike te ih koristeći ugrađene funkcije – poput curve() za izradu krivulje, transformirati u nasumično generirane oblike. Kako bi se oblici kreirali na pravilan i kontroliran način (kako slika ne bi postala previše apstraktna i izgubila osnovnu formu), potrebno je koristiti ugrađene funkcije pop() i push() unutar p5.js biblioteke.

Funkcija push() omogućava privremeno pohranjivanje trenutnog stanja transformacija, uključujući translacije, rotacije i druge promjene koje utječu na način crtanja što je potrebno jer se tijekom iteracije slikom primjenjuju različite transformacije na svaki piksel. Bez push() funkcije, svaka transformacija "stapala" bi se s prethodnom, što bi rezultiralo nepredvidivim efektima i mutnim crtežom. S druge strane, funkcija pop() vraća transformacijsko stanje na ono koje je bilo spremljeno pomoću push() funkcije te omogućava da se nakon svakog piksela, bez obzira na transformacije koje su primijenjene na njega, vrati na čisto i nepromijenjeno stanje za sljedeći piksel. Korištenje pop() i push() funkcija osigurava da svaki piksel bude obrađen u izoliranom okruženju, što je ključno za stvaranje koherentne i smislene slike.

Kako bi se generirani oblici međusobno razlikovali, osim korištenja funkcija za nasumičnost unutar kreacije same krivulje (nasumičnih koordinata kroz koje prolazi krivulja), koristit će se i funkcija rotate() koja svaku krivulju rotira ovisno o nasumično generiranim stupnjevima rotacije.

```
function draw() {
  if (img) {  //Provjera je li slika učitana
   for(let col=0;col<img.width;col+=densitySlider.value()){
     for(let row=0;row<img.height;row+=densitySlider.value()){
      push();

      let c = img.get(col,row);
      translate(col,row);
      rotate(radians(random(weightSlider.value()))  //Nasumična rotacija
      stroke(color(c));  //Boja oblika
      strokeWeight(random(strokeSlider.value()))  //Debljina oblika
      if(algorithSelect.value()=="Random"){
        curve(col,
            row,
            cos(col),
            sin(row)*random(30)*cos(row),
            cos(col)*random(30),
            sin(row)*random(30),
            cos(col)*random(30),
            sin(row)*random(30))
       }
       pop();
     }
 }
```

Isječak programskog koda 6. *Algoritam za stvaranje krivulja – p5.js*

Paper.js također koristi slične funkcionalnosti poput funkcija push() i pop() u p5.js, ali s različitim pristupom i sintaksom. U Paper.js stvaraju se staze (Path objekti) koje predstavljaju vektorske elemente, a transformacije se primjenjuju na te staze pomoću metoda, kao što su rotate() i translate(). Kako se za svaki element stvara zasebna staza, to dovodi do mogućnosti manipulacije svakog elementa zasebno - izoliranog od ostalih elemenata, no stvaranje zasebne staze za svaki element može dovesti do nagomilavanja velikog broja objekata kod kompleksnijih primjera i tako uvelike utjecati na potrošnju memorije i resursa. Također, zbog razlika u pristupu vektorskim i rasterskim elementima, u Paper.js-u nije moguće postići određene transformacije jer bi kreiranjem zasebnih staza samo izvođenje programa zahtijevalo previše resursa i značajno utjecalo na performanse programa, tako da se u konačnici radovi generirani u Paper.js-u (slika 9.) vizualno razlikuju od onih generiranih u p5.js alatu (slika 8.).

```
img.onload = function() {
        var raster = new paper.Raster(img);
        for (var x = 0; x < raster.width; x += 10) {
                for (var y = 0; y < raster.height; y += 10) {
                        var pixelColor = raster.getPixel(x, y);  // Dohvaćanje boje piksela
                        path = new paper.Path();  //Stvaranje nove putanje
                        var startPoint=new paper.Point(x,y);
                        path.moveTo(startPoint);
                        var handlePoint1 = new paper.Point(x + Math.random() * 160 , y
+ Math.random() * 50 - 25);
                       var handlePoint2 = new paper.Point(x + Math.random() * 27, y +
Math.random() * 70);
                       var endPoint = new paper.Point(x+ Math.random() * 50 , y +
Math.random() * 20);
                        var curveSegment = new paper.Segment(handlePoint1,
handlePoint2, endPoint);
```

```
                    path.add(curveSegment);  //dodavanje krivulje putanji
                    path.strokeColor = pixelColor;
                    path.strokeWidth =  Math.floor(Math.random() * 20);
                    path.strokeCap = 'round';
                    }
            }
            raster.remove();  // Ukloni raster objekt slike
    }
```

Isječak programskog koda 7. *Algoritam za stvaranje krivulja – Paper.js*

Slika 8. *Primjeri generiranih radova – p5.js*

Slika 9. *Primjeri generiranih radova – Paper.js*

3.2. Primjer vizualizacije podataka

U primjeru vizualizacije podataka p5.js će se testirati i usporediti s bibliotekom D3.js uz dodatak korištenja biblioteke Leaflet.js. Izradom interaktivne geografske karte koja prikazuje točke i područja interesa od učitanih datoteka te njihove međusobne relacije ispitat će se: mogućnosti alata da vizualiziraju i manipuliraju podatcima, podržanost različitih formata podataka, kompatibilnost rada s drugim vanjskim bibliotekama te implementacija interaktivnosti. Krajnji rezultat će biti program koji prikazuje kartu grada Zagreba i lokacije električnih punionica iz vanjske datoteke te ovisno o broju električnih utičnica, iscrtava granice gradskih četvrti skalirane u različitim bojama.

3.2.1. Vrste podataka u web okruženju

Biblioteke p5.js i D3.js raspolažu brojnim metodama za učitavanje i obradu vanjskih podataka te podržavaju sve potrebne vrste podataka, kako bi se primjer prikaza geografske karte i točaka interesa uspješno izradio. Tipovi podataka koji će se koristiti za vizualizaciju primjera su CSV(Comma-Separated Values), JSON(JavaScript Object Notation) i geoJSON.

CSV je tekstualni format koji se koristi za spremanje tabličnih podataka, poput podataka u tablicama baza podataka ili Excel datotekama. Podaci u CSV formatu organizirani su u redove, a svaki red sadrži vrijednosti različitih stupaca, koje su odvojene znakom zareza (,).

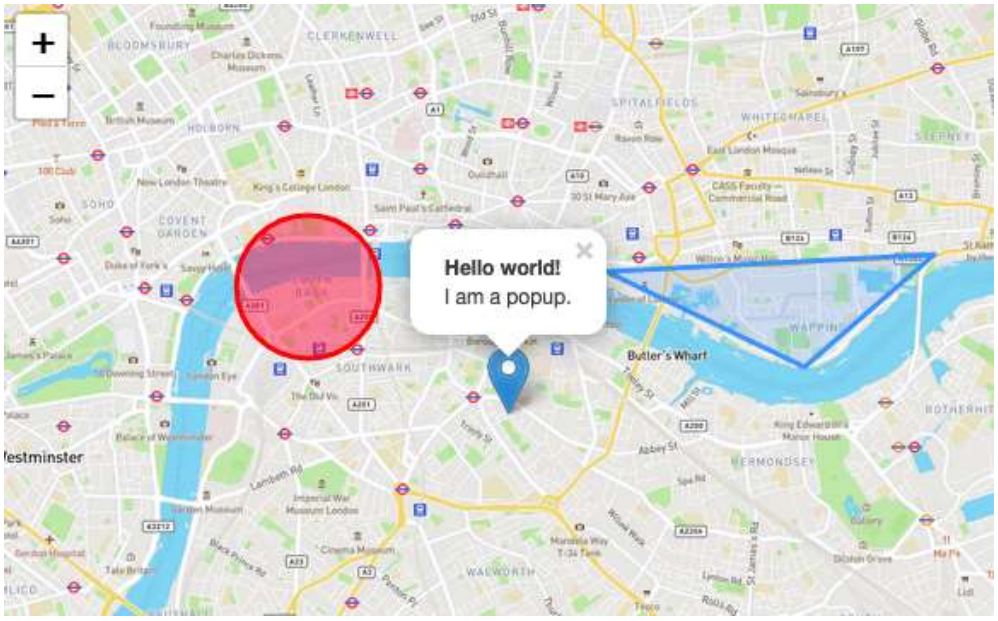
JSON je format za razmjenu podataka koji se temelji na JavaScript objektima. To je jednostavan i kompaktan format za prijenos podataka koji se često koristi u web razvoju za razmjenu podataka između klijenta i servera. JSON koristi sintaksu sličnu JavaScript objektima i nizovima, što ga čini jednostavnim formatom za čitanje i pisanje u web okruženju. GeoJSON je otvoreni format za prijenos geografskih podataka, baziran na JSON formatu. Koristi se za predstavljanje geografskih obilježja kao što su točke, linije i poligoni te povezanih atributa, kao što su nazivi i opisi geografskih elemenata. Za potrebe izrade primjera bit će korišten kao format za učitavanje informacija o granicama gradskih četvrti grada Zagreba(u obliku poligona).

3.2.2. Leaflet biblioteka za prikaz geografskih podataka

Leaflet je JavaScript biblioteka za interaktivno prikazivanje i rad s kartama u web okruženju. Razvijena s fokusom na jednostavnost i lakoću korištenja, Leaflet pruža razne funkcionalnosti za integraciju različitih vrsta karata i njihovu interakciju s korisnikom. Ova fleksibilna biblioteka omogućuje lako učitavanje različitih tipova kartografskih podataka, uključujući osnovne kartografske podloge, geografske oznake, linije i poligone.

Leaflet biblioteka također podržava integraciju s različitim kartografskim servisima i pružateljima podataka, omogućujući prikazivanje detaljnih karata s visokom razlučivošću. Kroz jednostavan API, programeri mogu kontrolirati funkcionalnosti i izgled karti, omogućujući jednostavno stvaranje prilagođenih i interaktivnih karata za različite potrebe.

U programiranju primjera Leaflet biblioteka će biti korištena kao osnovna kartografska podloga preko koje će biti izvođena daljnja vizualizacija i manipulacija podatcima.



Slika 10. *Primjer Leaflet karte*

3.2.3. Postavljanje početnih postavki prikaza geografske karte

Kako bi se kartografska podloga prikazala na platnu potrebno je stvoriti objekt koji će predstavljati Leaflet kartu te mu zadati početne postavke, kao što su početne koordinate, jačina uvećanja (engl. zoom) te vrsta kartografske podloge koja će biti korištena.

Stvaranje Leaflet objekta te implementacija njegovih postavki moguća je na dva načina, stvaranjem Leaflet objekta indirektno, uz pomoć dodatne Mappa.js biblioteke, koja olakšava crtanje i poboljšava kompatibilnost p5.js metoda na karti ili direktnom manipulacijom „L“ objekta koji predstavlja učitan Leaflet objekt u programu(u D3.js).

U prvom načinu (programiranom u p5.js) objekt Leaflet karte se stvara korištenjem metode „new Mappa('Leaflet')“ te mu se postavke (definirane u JSON obliku izravno u programu) apliciraju prilikom stvaranja karte u setup() dijelu p5.js programa. Kako bi karta bila prikazana na određenom Canvas elementu koristi se metoda overlay(), a kontrola iscrtavanja elemenata prilikom promjene položaja ili uvećanja karte (korisničkom interakcijom) odvija se pomoću metode onChange().

Drugi način (u D3.js) Leaflet objektu pristupa izravnom manipulacijom „L“ objekta, a dodavanje postavki pogleda (početnih koordinata i uvećanja) moguće je pomoću metode setView(), dok se odabir vrste kartografske podloge odvija ugrađenom Leaflet funkcijom tileLayer().

```
let mappa = new Mappa('Leaflet');
let options = {
  lat: 45.8150,
  lng: 15.9819,
  zoom: 12,
  style: "http://{s}.tile.osm.org/{z}/{x}/{y}.png"
}
function setup(){
  myMap = mappa.tileMap(options);
  myMap.overlay(canvas)
  myMap.onChange(callbackFunction);
}
```

Isječak programskog koda 8. *Postavljanje početnih postavki karte – p5.js*

```
<div id="map" style="width: 800px; height: 600px;"></div>

let map = L.map('map').setView([45.815010, 15.981919], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'.addTo(map);
```

Isječak programskog koda 9. *Postavljanje početnih postavki karte - D3.js*

3.2.4. Proces učitavanja podataka i pristup asinkronim funkcijama

Podatci potrebni za izradu primjera u program se učitavaju pomoću ugrađenih funkcija koje sadrže obje biblioteke.

U p5.js biblioteci podatci se učitavaju asinkronim funkcijama loadTable() – za čitanje CSV datoteka te njihovu pretvorbu u p5.Table objekte i loadJSON() - za čitanje JSON i GeoJSON datoteka. Asinkronost funkcija označava da se funkcije neće nužno završiti prije sljedeće linije koda pa je sve potrebne asinkrone funkcije za učitavanje podataka potrebno smjestiti u preload() dio p5.js programa koji osigurava izvršavanje funkcija prije setup() i draw() dijelova programa. Prilikom poziva loadTable() funkcije također je potrebno navesti lokaciju datoteke ili njen URL, tip tablice (u ovom slučaju CSV) te redak iz tablice u kojem su opisane oznake stupaca (u ovom slučaju "header"), dok je prilikom pozivanja loadJSON() funkcije potrebno navesti samo ime datoteke ili URL.

Prilikom učitavanja podataka u D3.js biblioteci koriste se statičke asinkrone metode sadržane unutar D3 objekta, za učitavanje CSV datoteka D3.csv() te za učitavanje JSON datoteka D3.json(). Za razliku od p5.js biblioteke D3.js pristupa asinkronom načinu učitavanja podataka putem JavaScript "obećanja" (Promise API) kako bi se osiguralo da učitavanje ne blokira izvršenje ostatka programa. Kada se učitavanje podataka dovrši, poziva se odgovarajuća funkcija (npr. .then()) s rezultatima učitavanja ili se obrađuje greška (npr. .catch()), ako se dogodi problem tijekom učitavanja.

```
function preload() {
  data = loadTable("./elektricne_punionice.csv", "csv", "header");
  boundary = loadJSON("./map.geo.json");
}
```

Isječak programskog koda 10. *Učitavanje CSV i GeoJSON podataka – p5.js*

```
d3.csv('./zagreb_points.csv').then(function (tocke) {
    d3.json('./map.geo.json').then(function (data) {
        //manipulacija i vizualizacija podataka
        });
});
```

Isječak programskog koda 11. *Učitavanje CSV i GeoJSON podataka – D3.js*

3.2.5. Dohvaćanje podataka iz učitanih datoteka

Kada je učitavanje podataka obavljeno, njihove vrijednosti je potrebno dohvatiti kako bi daljnja manipulacija bila moguća. U osnovi dohvaćanje podataka iz CSV i JSON datoteka je u p5.js i D3.js bibliotekama gotovo identično, zbog već ustaljene prakse dohvaćanja tabličnih podataka u skoro svim JavaScript bibliotekama i jezicima.

Kako bi dohvatili sve podatke iz CSV datoteke (za potrebe primjera iz datoteke koja sadrži sve točke interesa i njihove koordinate) potrebno je napisati petlju koja prolazi kroz sve redove tablične datoteke te ugrađenim funkcijama, kao što su to get() u p5.js ili jednostavnim navođenjem imena entiteta za dohvaćanje u D3.js, dohvatiti svaki redak zasebno te ga dalje manipulirati ili vizualizirati. Razlika je u sintaksi, ali koncept je isti - u oba slučaja, redovi su pohranjeni kao objekti te ih se može dohvatiti pomoću indeksa, a vrijednosti u redovima mogu se dohvatiti pomoću njihovih ključeva (naziva stupaca). Također, ovisno o krajnjem rezultatu i željenom tipu podataka moguća je potreba za pretvorbom podataka iz redaka tablice, koji su uvijek u tekstualnom obliku prilikom učitavanja, u neke druge oblike lakše za manipulaciju – što je moguće učiniti ugrađenom funkcijom „parse“.

Dohvaćanje vrijednosti iz učitanih JSON i GeoJSON nije specifično i ne ovisi o bibliotekama, već se koristi općeniti pristup dohvaćanja vrijednosti iz JavaScript objekata. U primjeru je korištena GeoJSON datoteka koja sadrži strukturu opisanu svojstvima (engl. features) gdje je svaki poligon (granice područja na karti) zapravo jedno svojstvo koje u sebi sadrži podatke o točkama kroz koje poligon prolazi (njihove koordinate) te dodatne informacije poput naziva, boje, oblika i sl.

```
tocke.forEach(function (tocka) {
        let tockaLon = tocka.X;
        let tockaLat = tocka.Y;
});
```

Isječak programskog koda 12. *Primjer dohvaćanja koordinata točaka – D3.js*

3.2.6. Pretvorba učitanih podataka u grafičke elemente

Kako bi u p5.js na karti bilo moguće vizualizirati, točke interesa učitane iz CSV datoteke njihove originalne koordinate (koordinate iz stvarnog svijeta) najprije trebaju biti pretvorene u koordinate platna, gdje svaka točka (piksel) platna predstavlja određenu koordinatu karte. Pretvorbu je moguće ostvariti Leaflet ugrađenom funkcijom latLngToPixel() koja unesenu geografsku širinu i dužinu pretvara u koordinate piksela na platnu, pomoću kojih je zatim moguće precizno vizualizirati točke interesa upotrebom ugrađenih alata za crtanje rasterskih oblika iz p5.js biblioteke. Prilikom svake promjene na karti (naknadnih pomicanja i uvećanja) potrebno je sve oblike iscrtavati nanovo kako bi njihova pozicija na karti ostala relevantna, što je moguće postići pozicioniranjem svih funkcija za vizualizaciju elemenata unutar povratne funkcije koja se aktivira prilikom svake promjene – koje detektira Leaflet funkcija onChange().

Biblioteka D3.js vizualizaciji podataka pristupa vektorski te je za potrebe vizualizacije elemenata na Leaflet karti potrebno instancirati zaseban SVG sloj, koji se dodaje na već postojeći Canvas element. Nakon što je SVG „nadsloj“ instanciran na njega je moguće ugrađenom funkcijom .append() dodati elemente (npr. krugove koji predstavljaju točke interesa) koji će se ponašati kao „dijete“ (engl. child element) glavnom SVG sloju. Svi glavni atributi (radijus, X i Y koordinate) postavljaju se funkcijom .attr(), a dodatni stilovi (npr. boja, prozirnost, stilizacija) dodaju se funkcijom .style(). Također zbog načina na koji D3.js pristupa svojim objektima - gdje je svaki element objekt za sebe, moguće je svakom objektu zasebno dodati različite atribute, stilove i funkcije interaktivnosti.

```
for (let i = 0; i < numRows; i++) {
  let la = myMap.latLngToPixel(lat[i], lng[i]);

  if (myMap.map.getBounds().contains({ lat: lat[i], lng: lng[i] })) {
    noStroke();
    ellipse(la.x, la.y, brUticnica[i] * 5, brUticnica[i] * 5);
  }
}
```

Isječak programskog koda 13. *Vizualizacija točaka interesa na karti – p5.js*

```
const svgLayer = L.svg();
svgLayer.addTo(map);
const svg = d3.select(map.getPanes().overlayPane).select('svg');
const g = svg.append('g').attr('class', 'leaflet-zoom-hide');

data.forEach(function(d) {
    let lat = parseFloat(d.Y);
    let lon = parseFloat(d.X);
     let radius = radiusScale(parseFloat(d.BROJ_UTICNICA));

    g.append('circle')
  .attr('class', 'circle')
  .attr('cx', map.latLngToLayerPoint([lat, lon]).x)
  .attr('cy', map.latLngToLayerPoint([lat, lon]).y)
  .attr('r', radius)
```

```
.style('fill', 'blue')
.style('fill-opacity', 0.5)
.on('mouseover', function(e) {
  d3.select(this).append('title').text(naziv);
})
.on('mouseout', function(e) {
  d3.select(this).select('title').remove();
                });
      });
});
```

Isječak programskog koda 14. *Vizualizacija točaka interesa na karti – D3.js*

3.2.7. Manipulacija podataka pomoću ugrađenih matematičkih funkcija

Nakon što su učitane datoteke koje sadrže koordinate točaka interesa i koordinate točaka koje čine poligon, potrebno je utvrditi njihove relacije.

Za potrebe detekcije nalazi li se određena točka interesa unutar granica poligona (gradskih četvrti) u p5.js potrebno je napisati vlastitu funkciju, koja za svaku koordinatu točke izračunava pripadnost određenom poligonu, jer p5.js biblioteka ne posjeduje ugrađenu funkciju koja bi olakšala problem prostorne relacije.

Biblioteka D3.js sadrži ugrađenu funkciju d3.polygonContains() koja kao ulazne parametre prima koordinate poligona (skup točaka) te koordinate točke kojoj se treba provjeriti pripadnost poligonu. Ako je točka unutar granica poligona, funkcija će vratiti povratnu informaciju boolean tipa „true“, a u suprotnom „false“.

```
function isTockaUnutarPoligona(x, y, poligon) {
 let unutar = false;
 let j = poligon.length - 1;
 for (let i = 0; i < poligon.length; i++) {
  if ((poligon[i][1] < y && poligon[j][1] >= y) || (poligon[j][1] < y && poligon[i][1] >= y)){
    if (poligon[i][0] + ((y - poligon[i][1]) / (poligon[j][1] - poligon[i][1])) *
          (poligon[j][0] - poligon[i][0]) <  x) {
      unutar = !unutar; } }
  j = i;
 }
 return unutar;
}
```

Isječak programskog koda 15. *Provjera pripadnosti točke poligonu - p5.js*

Kada je završeno određivanje relacija točaka interesa i poligona u kojima se nalaze, svaki poligon sadrži sumu točaka koje se nalaze unutar njegovih granica. Boja svakog poligona upravo će ovisiti o sumi svih pripadajućih točaka te će za pretvorbu sume u konkretnu boju biti korištene funkcije skaliranja koje sadrže obje biblioteke.

U p5.js biblioteci ulazni podatci mogu se mapirati koristeći ugrađenu funkciju map() koja preslikava vrijednosti iz jednog zadanog raspona u drugi. Ovaj postupak se također može primijeniti na osnovne komponente boje (crvenu, zelenu i plavu) kako bi poligoni sadržavali različite boje ovisno o ulaznim podacima. Osim funkcije za mapiranje p5.js ne raspolaže funkcijama za definiranje skala i kompleksnijih transformacija ulaznih parametara.

U D3.js biblioteci skaliranje ulaznih brojeva u boje se postiže definiranjem ugrađenih ljestvica boja (engl. color scales), koje preslikavaju kontinuirane ili diskretne vrijednosti u odgovarajuće boje. Najprije se definira ljestvica boja pomoću d3.scaleSequential() funkcije te se postavlja domena ljestvice prema rasponu ulaznih

brojeva. Zatim se poziva ljestvica boja s određenim ulaznim brojem kako bi se dobio odgovarajući nijansirani rezultat.

```
let colorScale = d3.scaleSequential()
  .domain([0, 50])  //Ulazni raspon (0-50)
  .interpolator(d3.interpolateViridis);


let mappedColor = colorScale(inputValue);
```

Isječak programskog koda 16. *Definiranje ljestvice boja – D3.js*

3.2.8. Implementacija interaktivnosti nad grafičkim objektima

U p5.js biblioteci interaktivnost se postiže praćenjem događaja (engl. events) poput akcija koje se zbivaju korištenjem računalnog miša (klikova, prijelaza, pomicanja) ili događaja koje uzrokuju ostali ulazni uređaji – poput računalne tipkovnice. Osnovne funkcije za praćenje događaja koje uzrokuje računalni miš su mouseX i mouseY za praćenje trenutnog položaja miša te funkcije mousePressed i mouseReleased za detekciju klika miša. Kako bi implementacija interaktivnosti na pojedine objekte, pomoću ugrađenih funkcija za detekciju pozicije miša bila moguća, potrebno je implementirati vlastite funkcije koje provjeravaju i uspoređuju položaj računalnog miša i trenutni položaj objekta, jer p5.js ne nudi mogućnost dodavanja „detektora događaja“ (engl. event listener) na vlastiti grafički objekt. Kako bi, primjerice, objekt ispisivao određen tekst prilikom klika miša (točka interesa prikazuje svoju adresu), algoritam za provjeru relacije između pozicije miša i objekta potrebno je ugnijezditi u mouseClick() globalnu funkciju koja će se pokrenuti svaki put kada detektira klik računalnog miša.

U D3.js, interaktivnost se postiže korištenjem vrlo širokog skupa funkcija za dodavanje i reagiranje na događaje. Primjenom .on() metode na SVG elemente moguće je implementirati interaktivnost kao što su „hover“ efekti, prikazivanje teksta pri prijelazu miša, mijenjanje stilova i atributa. Korištenjem ugrađenih događaja poput

'click', 'mouseover' ili 'mouseout', moguće je omogućiti korisnicima interakciju s grafičkim elementima ovisno o stanju ulaznog uređaja – primjerice računalnog miša. Takav pristup interaktivnosti D3.js čini puno intuitivnijim i sintaksom značajno jednostavnijim od p5.js biblioteke, gdje je za dodavanje dijelova koda koji reagiraju na pojedini događaj dovoljno koristiti ugrađenu funkciju i navesti tražen događaj, bez potrebe za programiranjem dodatnih funkcija koje povećavaju kompleksnost koda.

```
function mouseClick() {
  let lng = data.getColumn("X");
  let lat = data.getColumn("Y");
  let adresa = data.getColumn("ADRESA");
  for (let i = 0; i < numRows; i++) {
    let la = myMap.latLngToPixel(lat[i], lng[i]);
    if (
      mouseX < la.x + radius/2 &&
      mouseX > la.x – radius/2 &&
      mouseY > la.y - radius/2  &&
      mouseY < la.y + radius/2  ) {
      textSize(brUticnica[i] * 5);
      text(adresa[i], la.x - radius/2, la.y - radius/2);
    }
  }
}
```

Isječak programskog koda 17. *Dodavanje interaktivnosti – p5.js*

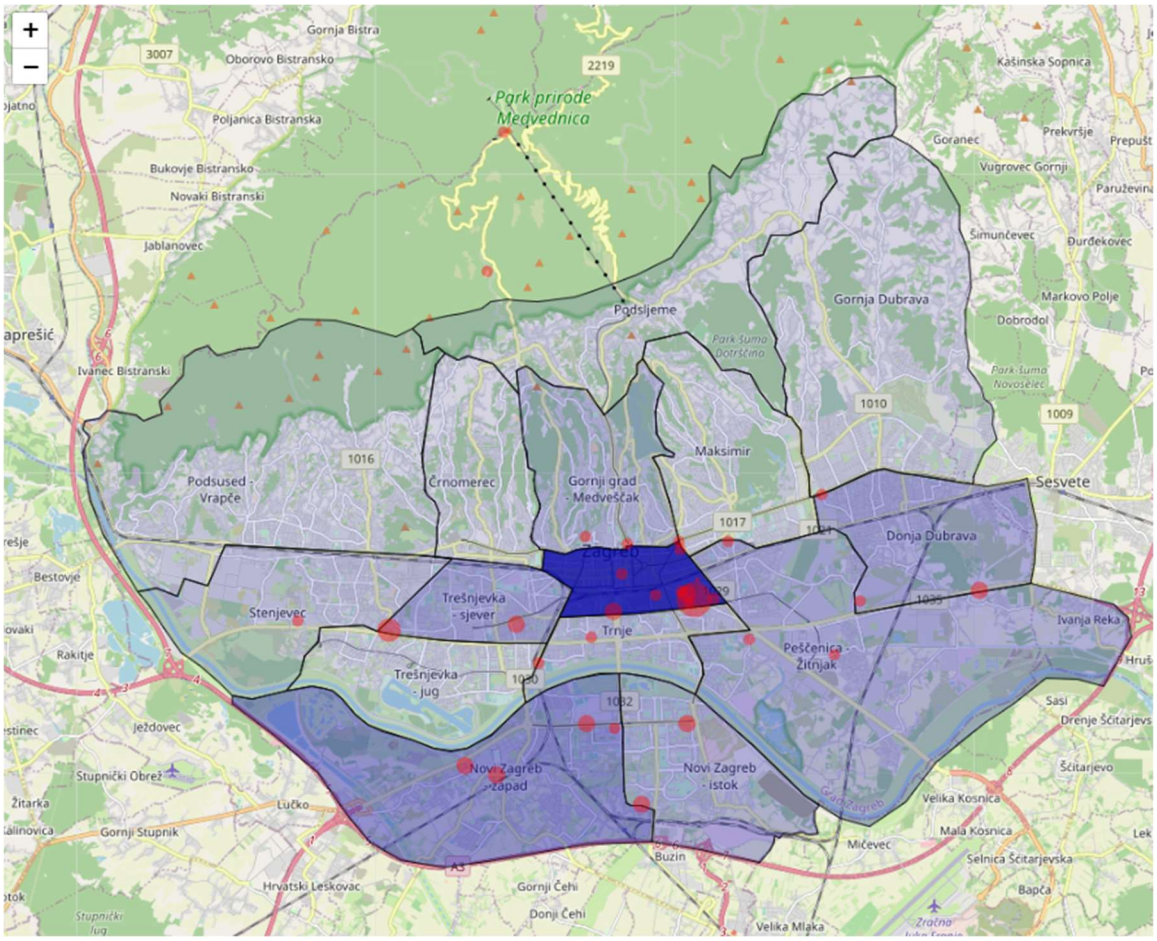
```
g.append('circle')
.attr('class', 'circle')
.attr('cx', map.latLngToLayerPoint([lat, lon]).x)
.attr('cy', map.latLngToLayerPoint([lat, lon]).y)
.attr('r', radius)
.on('mouseover', function(e) {  //Pomicanje miša unutar objekta
  d3.select(this).append('title').text(naziv);
})
.on('mouseout', function(e) {  //Pomicanje miša izvan objekta
  d3.select(this).select('title').remove();
});
  });
});
```

Isječak programskog koda 18. *Dodavanje interaktivnosti na objekt – D3.js*

3.2.9. Realizacija praktičnog primjera vizualizacije podataka

Nakon što je program povezan s vanjskom Leaflet bibliotekom i zadane su početne postavke prikaza geografske karte, vanjski se podatci učitavaju, dohvaćaju, obrađuju i manipuliraju. Ti podatci se uz pomoć ugrađenih funkcija za vizualizaciju rasterskih (p5.js) i vektorskih (D3.js) grafika prikazuju na Leaflet geografskoj karti. Na samom kraju, potrebno je implementirati funkcije interaktivnosti, kako bi adresa pojedinih točaka interesa bila vidljiva prilikom prelaska miša(slika 12.).

Kako obje biblioteke raspolažu s potrebnim ugrađenim funkcijama, kako bi primjer bio riješen na gotovo identičan način (uz značajne razlike u samom pristupu i sintaksi), podatci su prikazani na gotovo identične načine te ne postoji vidljiva vizualna razlika u finalnom rezultatu (slika 11.).



Slika 11. *Realiziran primjer vizualizacije podataka na geografskoj karti*

Slika 12. *Prikaz teksta prilikom prelaska računalnog miša*

4. Rasprava rezultata

4.1 Primjer generativne umjetnosti

Obje biblioteke su se pokazale kao kompetitivni alati za izradu generativne umjetnosti od učitanog slikovnog predloška te je primjer uspješno isprogramiran u oba slučaja. Najveća razlika dviju biblioteka je njihov pristup prikazu grafičkih elemenata, što čini daljnju razliku u njihovom odnosu prema općoj sintaksi, načinu na koji obrađuju slike te kompleksnosti crtanja osnovnih oblika. Funkcijski orijentirana p5.js sintaksa vrlo je intuitivna i početnički orijentirana, za razliku od Paper.js sintakse koja zahtjeva dodatno predznanje o objektno-orijentiranom programiranju i načinu na koji se manipulira vektorskim grafikama. Pri izradi zakrivljenih linija, za potrebe primjera u p5.js biblioteci, dovoljno je pozvati ugrađenu funkciju, dok Paper.js zahtjeva dodatne korake i detaljnije definicije zakrivljena linije uvođenjem segmenata.

Spomenuta razlika u pristupu prikazu grafičkih elemenata najviše se očituje prilikom rada sa slikama. Za potrebe primjera, rasterski način prikaza se pokazao kao puno bolji izbor za obradu i prikaz generiranih slika, upravo zbog mogućnosti manipulacije pojedinačnim pikselima i velike količine detalja koju bi vektorskim načinom bilo vrlo teško prikazati. Također, kako bi određena slika bila obrađena i od nje stvorena generativna umjetnost, Paper.js biblioteka zahtjeva dodatni korak transformacije slikovnog predloška iz rasterskog u vektorski oblik, što dodatno narušava performanse i povećava kompleksnost koda.

Prilikom same vizualizacije i transformacije grafičkih objekata (rotacija, translacija, dodavanje stilova i nasumičnosti), pristup p5.js biblioteke s ugrađenim funkcijama pop() i push(), osim što je vrlo jednostavan sintaksom, značajno olakšava čitanje koda jer jasno označava početak i kraj transformacija. Pristup Paper.js-a transformaciji i izolaciji pojedinačnih objekata je sintaksom zahtjevniji te je za svaku novu iteraciju potrebno instancirati novi objekt staze, što može značajno utjecati na performanse u slučajevima obrade slika velike rezolucije, no takav pristup također daje mogućnost aplikacije složenijih efekata i precizniju manipulaciju transformacije – uz više opcija manipulacije pojedinačnim dijelovima vektorskih elemenata.

Kao alat za izradu generativne umjetnosti p5.js se ističe, zbog velikog broja raznovrsnih ugrađenih funkcija za kreativno kodiranje, gdje za svaki aspekt izrade, od uređivanja slika do generiranja nasumičnih brojeva, nudi kompetitivan broj naredbi i ugrađenih funkcija. Također uz dodatne funkcije manipulacije DOM elemenata, p5.js daje mogućnost instantnog testiranja i postavljanja ulaznih parametara unutar samog JavaScript koda, što može značajno pridonijeti brzini testiranja programa i olakšati proces spajanja vanjskih skripti i HTML elemenata svim početnicima u programiranju. Paper.js ne posjeduje velik broj dodatnih funkcija izvan spektra same vizualizacije i manipulacije grafičkim elementima, pa tako ne nudi dodatne funkcije nasumičnosti ili DOM manipulacije, koja može uvelike pomoći u samom procesu kreativnog kodiranja.

Paper.js biblioteka pokazala se alatom koji zahtjeva mnogo više uloženog vremena za potpuno razumijevanje svih pojmova i funkcija potrebnih za izradu vektorske grafike, a zbog manjka aktivne zajednice vrlo je teško pronaći dodatna pojašnjenja i pokazne primjere. Za razliku od p5.js biblioteke kojoj je primjena raznovrsna Paper.js u fokusu ima jedno područje – vektorske grafike te zbog toga obiluje kompleksnijim metodama koje pomažu u svakom koraku izrade i obrade vektorskih grafika.

Svakako, od ova dva alata, p5.js ima manje strmu krivulju učenja te je zbog svog funkcijski orijentiranog pristupa lakši za učenje i djeluje intuitivnije za svladavanje, dok također obuhvaća sve važnije koncepte JavaScript programiranja, kao što su kontrolne strukture, rad s funkcijama, manipulacija DOM elemenata, korištenje matematičkih operacija te stvaranje objekata i upoznavanje s tipovima podataka. Zbog mnogo aktivnije i šire zajednice brojnih GitHub projekata i YouTube lekcija te jednostavnog web-orijentiranog programskog sučelja, p5.js se istaknuo kao puno bolji izbor za početak učenja JavaScript sintakse i kao dobar uvod u kompleksnije programske jezike, dok se Paper.js pokazao kao alat za iskusnije programere koji su upoznati s osnovnim konceptima vektorskih grafika i kojima je važna mogućnost izrade kompleksnijih vektorskih grafika u web okruženju.

Alat	p5.js	Paper.js
Odabrani model prikaza	4,0	3,8
Organizacija koda	3	5
Jednostavnost sintakse	5	3
Rad sa slikama	4	3
DOM manipulacija	4	1
Funkcije nasumičnosti	4	1
Kompleksnije funkcije	2	5
Zajednica	5	3
Dokumentacija	5	3
Mogućnost učenja	5	3
Ukupna ocjena	**4,1**	**3,08**

***Tablica 2**. Tablica ocjena – primjer generativne umjetnosti*

4.2 Primjer vizualizacije podataka

Obje biblioteke su se pokazale kao kompetitivni alati za vizualizaciju učitanih podataka na geografskoj karti te je primjer uspješno isprogramiran u oba slučaja, no za razliku od primjera izrade generativne umjetnosti p5.js biblioteka se pokazala mnogo ograničenijim alatom u području vizualizacije i manipulacije podatcima.

Kompatibilnost s Leaflet bibliotekom za prikaz geografskih karata se pokazala boljom u slučaju p5.js alata, isključivo zbog njegove rasterske prirode (na kojoj je bazirana i Leaflet biblioteka), dok je u slučaju D3.js biblioteke potrebno učiniti dodatne korake, kako bi se SVG elementi vizualizacije mogli prikazati na samoj karti.

Učitavanje podataka kod obje biblioteke se pokazalo vrlo intuitivnim i jednostavnim procesom, no glavne razlike se očituju u pristupu alata prema korištenju asinkronih funkcija. D3.js prema asinkronim funkcijama pristupa pomoću ugrađenog „Promise API“ koji nam nudi mogućnost izvršavanja različitih radnji u slučaju nemogućnosti učitavanja podataka i kontrolu u ovisnosti o trenutnom stanju učitavanja podataka (pending, fullfilled, rejected), dok p5.js u ne nudi takav ili sličan mehanizam, već je sve asinkrone funkcije potrebno postaviti u preload() dio programa.

Kada su podatci učitani i dohvaćeni, predstoji vizualizacija podataka, koja se među bibliotekama razlikuje u općem pristupu vizualizaciji elemenata. Zbog već spomenute razlike u pristupu grafičkim elementima – rasterskog(p5.js) i vektorskog(D3.js), drugačiji je i pristup kojim se stvaraju grafički objekti poput krugova i poligona. Svakako, zbog veće kompleksnosti rasterskih oblika i SVG formata, način na koji D3.js vizualizira elemente zahtjeva konfiguraciju dodatnih slojeva te dodatnu manipulaciju nad platnom na kojem se prikazuje geografska karta, no nakon početne konfiguracije D3.js nudi vrlo intuitivan pristup kreiranju objekata iz učitanih podataka(koordinata) te zbog objektno-orijentiranog pristupa posjeduje mogućnosti poput instantnog dodavanja atributa, stilova i interakcije na svaki objekt zasebno. Također D3.js ne zahtjeva korištenje petlji, kako bi se oblici poput poligona (koji se sastoje od polja točaka) mogli vizualizirati, već je u samu ugrađenu funkciju, koja vizualizira objekt, moguće učitati cijelo polje koordinata – bez potrebe za iteracijom, što uvelike pridonosi poboljšanju performansi programa.

U području manipulacije podatcima D3.js se posebice istaknuo kao kompetentan alat s velikim brojem dodatnih matematičkih funkcija koje značajno pojednostavljuju matematičke procese (poput izračuna relacije točaka i poligona), što i nije iznenađujuće s obzirom da je naspram p5.js biblioteke kojoj je glavni imperativ raznolikost i jednostavnost, glavni fokus D3.js-a rad s podatcima i matematičkim problemima. p5.js se oslanja na već ugrađene funkcionalnosti unutar JavaScript-a te osim funkcija za mapiranje vrijednosti iz jedne domene u drugu, ne sadrži značajan broj matematičkih funkcija, koje bi pojednostavile proces manipulacije podatcima.

Dodavanje interaktivnosti na elemente i manipulacija događajima (engl. events), također je jedna od velikih prednosti D3.js alata naspram p5.js biblioteke, koja, iako sadrži značajan broj ugrađenih funkcija za detekciju, manipulaciju i rukovanje dolaznim događajima (pogotovo uzrokovanih računalnim mišem), zbog svoje funkcijski orijentirane sintakse, ne sadrži mogućnosti dodavanja interaktivnosti na zasebne elemente i grafičke objekte. D3.js svojim objektno-orijentiranim pristupom nudi mogućnosti, poput direktne aplikacije interaktivnosti na SVG grafičke objekte, što značajno pridonosi manjoj kompleksnosti koda i većoj razumljivosti samog rukovanja ulaznim događajima.

U konačnici, p5.js, iako nije alat kojemu je glavni fokus vizualizacija i manipulacija podataka, uspijeva realizirati zadani primjer sadržavajući dovoljan broj naredbi za

učitavanje, dohvaćanje i vizualizaciju, pritom zahtijevajući dodatne korake i veću kompleksnost koda nego je to očekivano, jer upravo jednostavnost same p5.js sintakse i njen pristup (funkcijski) zadaju poteškoće podatkovno kompleksnijim primjerima, koji zahtijevaju kompleksnije matematičke operacije i manipulacije.

Biblioteka D3.js svakako se istaknula kao puno bolji izbor za realizaciju zadatka vizualizacije podataka te iako rukovanje samim alatom zahtijeva prethodno poznavanje SVG grafičkih objekata i objektno-orijentiranog pristupa, u konačnici D3.js zahtijeva znatno manje kompleksnih radnji, zbog bogate baze ugrađenih funkcija koje pojednostavljuju svaki korak izrade, od samog pristupa asinkronim funkcijama, prilikom učitavanja podataka pa do aplikacije interaktivnosti. Iako je sintaksa i pristup p5.js biblioteke na prvi pogled jednostavnija, u procesu vizualizacije podataka krivulja učenja je svakako strmija naspram D3.js biblioteke, koja za svaki korak i proces vizualizacije sadrži opširnu dokumentaciju i veliku bazu primjera na raspolaganju, dok je u slučaju p5.js očit fokus aktivne zajednice na olakšanju samog procesa kreiranja grafičkih elemenata, crteža i animacija, a ne toliko vizualizacije i manipulacije podatcima.

Alat	p5.js	D3.js
Podržani formati podataka	4	5
Učitavanje podataka	3	5
Dohvaćanje podataka	5	5
Kompatibilnost s vanjskom bibliotekom	5	4
Manipulacija podataka	2	5
Vizualizacija podataka	4	5
Implementacija interaktivnosti	3	5
Zajednica	4	5
Kompleksnost koda	2	4
Mogućnost učenja	3	4
Ukupna ocjena	**3,5**	**4,7**

***Tablica 3**. Tablica ocjena – primjer vizualizacije podataka*

5. Zaključak

Velikim brojem raznovrsnih funkcija za kreativno kodiranje, relativno jednostavnom sintaksom i razumljivom dokumentacijom, p5.js biblioteka nudi dobar omjer između jednostavnosti i mogućnosti, što ju čini kompetitivnim alatom za sve koji se žele kreativno izraziti, a da pritom ne trebaju posjedovati mnogo predznanja. U sklopu kompleksnijih projekata upravo jednostavnost p5.js sintakse dolazi do izražaja kao jedna od mana, zbog načina na koji se odnosi prema organizaciji programskog koda, nedostatku naprednijih matematičkih funkcija i lošijih performansi prilikom korištenja zahtjevnijih algoritama.

Oba praktična primjera su pomoću p5.js biblioteke uspješno realizirana te se, usporedbom s konkurentnim alatima u zadanim područjima, p5.js pokazala kao puno kompetitivniji alat u kontekstu izrade generativne umjetnosti, gdje je zbog svoje posebice dobre dokumentiranosti, pristupa grafičkim objektima i raznolikosti funkcija za kreativno kodiranje zaslužila značajno bolju konačnu ocjenu(4,1) naspram svoje konkurencije(3,08). U području vizualizacije i manipulacije podataka p5.js biblioteka se pokazala sposobnim alatom, ali s velikim nedostatcima u pogledu kompleksnijih funkcija, rukovanju složenijim matematičkim problemima te nedovoljno intuitivnom aplikacijom interaktivnosti pa je za razliku od svoje konkurencije u navedenom području zaslužila značajno manju konačnu ocjenu(3,5 naspram 4,7).

U konačnici p5.js biblioteka se pokazala iznimno poticajnim alatom za razvoj tehnološke pismenosti i korištenje u šire obrazovne svrhe, upravo zbog svoje intuitivne i razumljive funkcijski orijentirane sintakse, imperativa na vizualno stvaralaštvo, konstruktivne dokumentacije bogate primjerima i velike raznolikosti funkcija koje mogu olakšati svaki korak učenja programiranja svim početnicima ili umjetnicima željnim kreativnog izražavanja te je svakako jedan od kompetitivnijih alata za učenje programiranja i programske sintakse u suvremenom web okruženju.

6. Literatura

1. Joke Voogt and Natalie Pareja Roblin. A comparative analysis of international frameworks for 21st century competences: Implications for national curriculum policies. Journal of Curriculum Studies, 44(3):299–321, 2012. doi:10.1080/00220272.2012.668938

2. Peppler, K.A., & Kafai, Y.B. (2005). Creative Coding : Programming for Personal Expression.

3. I. Greenberg, Processing: Creative coding and computational art, 1st ed. New York, NY: APRESS, 2008.

4. M. C. Mitchell and O. Bown, “Towards a creativity support tool in processing,” Nov. 2013, doi: https://doi.org/10.1145/2541016.2541096.

5. A. M. Noll, “The Beginnings of Computer Art in the United States: A Memoir,” Leonardo, vol. 27, no. 1, p. 39, 1994, doi: https://doi.org/10.2307/1575947.

6. B. Harvey, Computer science LOGO style. Vol. 1 Symbolic computing. Cambridge, Mass. ; London: Mit, 1997.

7. J. Maeda, Design by numbers. Cambridge, Mass.: Mit Press, 2001.

8. T. Terroso and M. Pinto, “Programming for non-programmers: An approach using creative coding in higher education,” 2022.

9. Tim, “What is creative coding?,” tim rodenbröker creative coding, https://timrodenbroeker.de/what-is-creative-coding/ [Pristupano: 19.06.2023.].

10. E. Arslan, Learn JavaScript with p5.Js: Coding for visual learners, 1st ed. Berlin, Germany: APress, 2018.

11. Boden, M. A., & Edmonds, E. A. (2009). What is generative art? Digital Creativity, 20(1-2), 21–46. doi:10.1080/14626260902867915

12. Srinivasan, R., & Uchino, K. (2021). Biases in Generative Art. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. doi:10.1145/3442188.3445869

13. Maeda, J. (2004). Creative Code. Thames & Hudson.

14. Bergstrom, Ilias and R. Beau Lotto. "Harnessing the Enactive Knowledge of Musicians to Allow the Real-Time Performance of Correlated Music and Computer Graphics." Leonardo, vol. 42 no. 1, 2009, p. 92-93. Project MUSE muse.jhu.edu/article/258997.

15. Bergstrom, I., & Lotto, R. B. (2015). Code Bending: A New Creative Coding Practice. Leonardo, 48(1), 25–31. doi:10.1162/leon_a_00934

16. K. Healy, Data visualization: A practical introduction. Princeton, NJ: Princeton University Press, 2018.

17. J. J. Sylvia IV, “Code/art approaches to data visualization,” in Research Methods for the Digital Humanities, Cham: Springer International Publishing, 2018, pp. 211–231.

18. C. Griffiths, “Computational visualization for critical thinking,” J. Sci. Technol. Arts, vol. 11, no. 2, pp. 9–17, 2019.

19. J. H. Sarachan, “Programming politics: Using p5.Js to create interactive art connected to current events,” in Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 2019.

20. “100R — orca,” 100r.co. https://100r.co/site/orca.html [Pristupano: 14.05.2023.].

21. “Processing.org,” Processing.org, 2019. https://processing.org/ [Pristupano: 14.05.2023.].

22. “Python Mode for Processing,” py.processing.org. https://py.processing.org/ [Pristupano: 14.05.2023.].

23. “JRubyArt (Code as Art),” ruby-processing.github.io. https://ruby-processing.github.io/JRubyArt/ [Pristupano: 14.05.2023.].

24. “p5.js | home,” P5js.org, 2019. https://p5js.org/ [Pristupano: 14.05.2023.].

25. Emil Sandberg. (2019). Creative Coding on the Web in p5.js. Blekinge Institute of Technology.

26. A. M. Mcnutt, A. Outkine, and R. Chugh, “A study of editor features in a creative coding classroom,” in Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, 2023.

27. M. Bostock, “D3.js - Data-Driven Documents,” *D3js.org*, 2013. https://d3js.org/

28. JainAbhijit, “Data visualization with the D3.JS Javascript library,” Journal of Computing Sciences in Colleges, Dec. 2014, doi: https://doi.org/10.5555/2667432.2667451.

7. Prilozi

Cjelokupni programski kod aplikacije nalazi se na CD disku.