Šolski center Novo mesto  
Srednja elektro šola in tehniška gimnazija  
Šegova ulica 112  
8000 Novo mesto

**SPLETNA STRAN ZA UREJANJE FRAKTALOV**(Seminarska naloga)

Predmet: Računalništvo  
  
Avtor: Jure Vajs, T4B  
Mentor: Gregor Mede, univ. dipl. inž. rač. in inf.

Novo mesto, april 2022

# POVZETEK IN KLJUČNE BESEDE

V seminarski nalogi sem opisal izdelavo spletne strani za urejanje fraktalov, ki sem jo izdelal ko zaključni izdelek zamaturo, in teorijo za fraktali, ki sem jih izdelal. To temo sem si izbral zaradi mojega zanimanja za fraktale, in želji po izdelavi le-teh. V prvem delu seminarske sem na kratko opisal vse uporabljene programske jezike, njihov pomen v seminarski nalogi, ter razložil in opisal matematično teorijo za vsakim izmed fraktalov. V drugem delu seminarske naloge pa sem opisal samo izdelavo spletne strani, ki sem jo razdelil na backend spletne strani, frontend spletne strani, ter postopek objave spletne strani na internet. Za lažjo predstavo izdelka sem v seminarsko nalogo veliko slik spletne strani, in pa same programske kode, ki spletno stran sestavlja.

Ključne besede:

* Fraktali
* Spletna stran
* P5.js
* JavaScript
* HTML

# KAZALA

## KAZALO VSEBINE

[POVZETEK IN KLJUČNE BESEDE 3](#_Toc100441375)

[KAZALA 4](#_Toc100441376)

[KAZALO VSEBINE 4](#_Toc100441377)

[KAZALO SLIK 5](#_Toc100441378)

[KAZALO TABEL 6](#_Toc100441379)

[1 UVOD 1](#_Toc100441380)

[2 TEORETIČNI DEL 2](#_Toc100441381)

[2.1 HTML IN CSS 2](#_Toc100441382)

[2.2 JAVASCRIPT 2](#_Toc100441383)

[2.3 P5.JS 2](#_Toc100441384)

[2.4 TEORIJA ZA FRAKTALI 3](#_Toc100441385)

[2.4.1 FRAKTALNO DREVO 3](#_Toc100441386)

[2.4.2 KOCHOVA KRIVULJA 3](#_Toc100441387)

[2.4.3 MANDELBROTOV NIZ 3](#_Toc100441388)

[3 PRAKTIČNI DEL 4](#_Toc100441389)

[3.1 SINTAKSA 4](#_Toc100441390)

[3.1.1 JAVASCRIPT 4](#_Toc100441391)

[3.1.2 P5.JS 4](#_Toc100441392)

[3.2 PROGRAMIRANJE FRAKTALOV 5](#_Toc100441393)

[3.2.1 DREVO 5](#_Toc100441394)

[3.2.2 KOCHOVA RAVNINA 7](#_Toc100441395)

[4 ZAKLJUČEK 9](#_Toc100441396)

[5 VIRI IN LITERATURA 1](#_Toc100441397)

[6 STVARNO KAZALO 2](#_Toc100441398)

[7 PRILOGE 3](#_Toc100441399)

## KAZALO SLIK

## KAZALO TABEL

# UVOD

V modernem času so spletne strani, najlažji in najbolj učinkovit način za prikaz podatkov ali vsebine, ki je dostopen vsem uporabnikom interneta. Spletno stran lahko ustvari in vzdržuje posameznik, skupina, podjetje ali organizacija za različne namene.

V tej seminarski nalogi bom izdelal, in nato opisal postopek izdelave spletne strani. Cilj naloge je izdelati čim bolj uporabnikom prijazno spletno stran, in se preizkusiti v znanju algoritmov in matematike pri izdelavi različnih fraktalov. Pri izdelavi strani se bo najverjetneje pojavilo veliko problemov, s katerimi se bom soočil z sojim znanjem, in pa različnimi viri iz interneta in knjig.

V prvem delu bom na kratko opisal uporabljene programske jezike in teorijo za izdelanimi fraktali, v drugem delu pa se bom poglobil v samo izdelavo spletne strani, ki jo bom podprl z izseki kode, ki sestavljajo spletno stran.

# TEORETIČNI DEL

Teoretični del seminarske naloge sestavlja izbira primernih programskih jezikov za izdelavo izdelka in spoznavanje z njimi, ter raziskovanje, spoznavanje in razumevanje fraktalov, ki bodo izdelani.

## HTML IN CSS

HTML in CSS sta jezika, ki skrbita za izgled in postavitev spletne strani. HTML ali Hyper Text Markup Language je [označevalni](https://sl.wikipedia.org/w/index.php?title=Ozna%C4%8Devalni_jezik&action=edit&redlink=1) [jezik](https://sl.wikipedia.org/wiki/Jezik_(sredstvo_sporazumevanja)) pri katerem se preko značk in atributov, izdelujejo [spletne](https://sl.wikipedia.org/wiki/Splet) strani. Predstavlja osnovo spletnega dokumenta. Poleg prikaza dokumenta v [spletnem brskalniku](https://sl.wikipedia.org/wiki/Spletni_brskalnik) se z njim hkrati določi tudi zgradba in semantični pomen delov dokumenta. ([HTML - Wikipedija, prosta enciklopedija (wikipedia.org)](https://sl.wikipedia.org/wiki/HTML)). CCS ali Cascading Style Sheets, pa je slogovni jezik, ki skrbi za reprezentacijo spletnih strani. Z njim lahko določamo barve, velikosti, odmike, pozicije, in vrsto drugih atributov. CSS nam omogoča zmanjšanje ponavljanja kode, saj omogočimo množici strani uporabo istih podlog, kar lahko bistveno zmanjša njihovo velikost.

## JAVASCRIPT

JavaScript je objektni programski jezik, ki je bil razvit z namenom, da bi spletnim razvijalcem pomagal pri ustvarjanju spletnih strani. JavaScript lahko sodeluje s HTML kodo in s tem poživi stran z dinamičnim izvajanjem. Podpirajo ga vsi novejši spletni brskalniki. Sintaksa jezika ohlapno sledi programskemu jeziku C.

## P5.JS

P5.js je JavaScript knjižnica, ki se uporablja za kretivno kodiranje, s poudarkom na tem, da je koda dostopna in vključujoča za umetnike in oblikovalce. Temelji na okolju za ustvarjalno programiranje Processing, njegov namen pa je narediti programski jezik bolj prijazen uporabniku s pomočjo vizualizacije. P5.js programi se na splošno imenujejo skice (angl. sketches), saj programerjem omogočajo , da hitro ustvarijo prototipe programske opreme, ali da preizkusijo novo idejo. Vendar pa uporabnik ni omejen samo na risanje na platno (angl. canvas), saj lahko kot svojo skico obravnava celotno spletno stran s pomočjo HTML5 objektov za besedilo, vnos, zvok in spletno kamero.

## TEORIJA ZA FRAKTALI

Izraz "fraktal" je skoval matematik [Benoit Mandelbrot](https://en.wikipedia.org/wiki/Beno%C3%AEt_Mandelbrot) leta 1975. Mandelbrot ga je utemeljil na latinskem [*fractus*](https://en.wiktionary.org/wiki/fractus#Latin), kar pomeni "zlomljen".([Fraktal - Wikipedia](https://en.wikipedia.org/wiki/Fractal)) Fraktali so neskončni samopodobni vzorci ali geometrijske obloke, ki so ustvarjene s ponavljanjem preprostega posopka znova in znova v stalni povratni zanki. Fraktalni vzorci so zelo poznani, saj je narava polna fraktalov kot so drevesa, reke, obale, oblaki, školjke, rastline. Nekateri fraktali obstajajo le iz umetniških razlogov, drugi pa so zelo uporabni. Fraktali so zelo učinkovite oblike za radijske antene in se uporabljajo v računalniških čipih za učinkovito povezavo vseh komponent.

### FRAKTALNO DREVO

Fraktalno drevo je eden izmed najpreprostejših primerov fraktala. Definirano je rekurzivno s simetričnim razvejanjem, kar pomeni, da je opredeljeno glede na sebe (prejšnjo generacijo). Vsaka generacija ima dolžino, ki je nek procent dolžine njene predhodne generacije (dolžina nesme biti enaka ali daljša kot njena predhodna dolžina). Edina izjema je deblo drevesa, ki ima podano dolžino.

### KOCHOVA KRIVULJA

Kochova krivulja je eden izmed najzgodnejših fraktalov, ki se je prvič pojavil v prispevki švedskega matematika Helgea von Kocha iz leta 1904. Sestavimo jo lahko s tremi koraki. Najprej razdelimo odsek črte na tri enake dele, nato narišemo enakostranični trikotnik, ki ima za osnovnico srednji del na razdeljenem segmentu, in je obrnjen navzven, na koncu pa odstranimo del, ki je osnovnica trikotnika.

### MANDELBROTOV NIZ

Mandelbrotov niz je množica kompleksnih števil c, za katera funkcija f{c}(z) = z^2 + c ne zdrsne v neskončnost. Števila ki ne zdrsnejo v neskončnost predstavljajo orednji del niza. Števila ki zdrsnejo, pa lahko obarvamo po različnih sistemih barvanja, ki jim pripišejo barvno vrednost glede na število iteracije, pri kateri so zdrsnila v neskončnost. Mandelbrotov niz izvira iz področja kompleksne dinamike, ki sta ga na začetku 20. Stoletja raziskovala francoska matematika Pierre Fatou in Gaston Julia. Ta fraktal sta leta 1978 prvič opredelila in narisala Robert W. Brooks in Peter Matelski kot del študije Kleinianovih skupin. Matematično preučevanje Mandelbrotovega niza se je v resnici začelo z delom matematikov Adriena Douadyja in Johna H. Hubbarda, ki sta postavila številne njegova temeljne lastnosti in množico poimenovala v čast Mandelbrota za njegovo vplivno delo na področju fraktalne geometrije. Mandelbrotov niz je postal priljubljen zunaj matematike tako zaradi svoje estetske privlačnosti kot primera kompleksne strukture.

# PRAKTIČNI DEL

Za izdelavo fraktalov sem se odločal med Programskim jezikom Processing in pa p5.js, ki je knjižnica vgrajena v programski jezik JavaScript, in je bila narejena po sledeh Processinga. Na koncu sem se odločil za p5.js, saj mi je omogočal programiranje v okolju Vscode, kar je pomenilo da sem imel vse datoteke v isti mapi, in pa lažjo vključitev p5.js skic v spletno stran. Po tem, ko pa sem se odločil za programske jezike, ki so bili po mojem mnenju najbolj primerni za izdelavo izdelka in po tem ko sem se spoznal z fraktali, sem začel z izdelavo splene strani.

## SINTAKSA

### JAVASCRIPT

Ker sem vse skice pisal po p5.js strukturi programa, sem rabil le tiste primarne JavaScript ukaze, ki so mi omogočali branje vrednosti iz vnosnih mest in drsnikov, in pa ukaz ki omogoča vpise določenih vrednosti v konzolo.

|  |  |
| --- | --- |
| **document.getElementBy().value** | Prebere vrednost gumba po njegovem ID-ju,  in ga pripiše določeni spremenljivki. |
| **console.log()** | Vpiše vrednost spremenljivke med oklepaji v konzolo. |

### P5.JS

Kljub temu da je p5 knjižnica JavaScripta, ima zaradi svoje grafične naklonjenosti nekoliko drugačno sestavo programa. Vsak program je sestavljen iz funkcije setup(), ki se izvede prva ob zagonu programa, a le enkrat, in funkcije draw(), ki se izvede za funkcijo setup, in se ponavlja neprestano.

|  |  |
| --- | --- |
| **function setup()** | Pokliče se le enkrat, in sicer ob zagonu programa. Uporablja se za deifiniranje začetnih lastnosti skice |
| **function draw()** | Klicana neposredno po setup() funkciji. Deluje neprekinjeno do ustavitve programa ali do klica noLoop() funkcije |

Ostale funkije ki niso primarne, a sem jih uorabil pri vseh skicah:

|  |  |
| --- | --- |
| **windowResized()** | Se izvede takrat, ko se spremeni višina ali širina spletne strani |
| **resizeCanvas()** | Omogoča spreminjanje velikosti platna. Večinoma uporabljena znotraj windowResized() funkcije |
| **CreateCanvas()** | Ustvari platno z podano širino in višino |
| **canvas.parent()** | Znotraj oklepajev se vpiše id HTML div-a, preko katerega lahko lažje prilagajamo pozicijo in lastnosti skice na strani |
| **angleMode()** | Nastavi mersko enoto za kote. Možnosti izbire med radiani in stopinjami |
| **translate()** | Prestavi izhodiščno točko risanja levega zgornjega kota na željeno točko |
| **rotate()** | Zavrti črto ali lik za določeno vrednost |
| **push()** | Shrani trenutne nastavitve sloga, risanja in transformacije |
| **pop()** | Shranjene nastavitve iz push() funkcije obnovi. Uporablja se vedno v kombinaciji z funkcijo push() |

## PROGRAMIRANJE FRAKTALOV

Vseh fraktalov sem se načeloma lotil na isti način. Najprej sem si prebral članke in teorijo o določenem fraktalu, nato pa sem po svojih najboljših močeh poskusil ta fraktal sprogramirati sam. Ko sem naletel na kakšen problem ali zanko, ki je nisem znal sam razvozlati, sem se obrnil na pomoč virov iz interneta, ki so mi močno pomagali.

### DREVO

#### Z UREJANJEM

Ker je bil ta fraktal najbolj preprost izmed treh, sem se ga lotil prvega. Po tem ko, je v funkciji setup podana širina, višina in pa odzadje platna, program bere uporabnikov vnos za kot alfa in kot beta drevesa. Kot alfa predstavlja odmik desne veje glede na osnovnico, kot beta pa odmik leve veje glede na osnovnico.

alpha = document.getElementById("kotalfa").value;

console.log(alpha);

beta = document.getElementById("kotbeta").value;

console.log(beta);

Za tem sem z funkcijo translate poiskal sredino spodnje stranice platna, in podal barvo in debelino risanja.

translate(windowWidth/6,windowWidth/4);

stroke (255);

strokeWeight(4);

Nato sem klical funkcijo draw2(), pri kateri se je začelo risanje drevesa. Glede na podano dolžino pri klicanju, je funkcija narisala iz izhodišča (ki je bilo v prejšnem koraku spremenjeno) črto, in s funkcijo translate() prestavila izhodišče na zgornji konec narisane črte. Nato gre funkcija v if zanko, v kateri se razdeli s funkcijama push() in pop() na dve veji. Pri vsaki izmed vej se najprej pot zavrti za določeni kot (pri vsaki je ta kot drugačen), nato pa se funkcija draw2() kliče sama v sebi z dolžino, ki je nek delež stare dolžine, in ustvari neskončno zanko, ki pa se zaradi if stavka izvaja le dokler je dolžino nad neko vrednostjo.

function draw2(lenght){

    line(0,0,0,-lenght);

    translate(0,-lenght);

    if(lenght>9)

    {

        push();

        rotate(alpha);

        draw2(lenght \* 0.66);

        pop();

        push();

        rotate(beta);

        draw2(lenght \* 0.66);

        pop();

    }

}

#### NA ZAČETNI STRANI

Zaradi majhne računske zahtevnosti tega fraktala, je bil najbolj primeren za prikaz na začetni strani, ki bi vključeval zelo hitro in zanimivo interaktivnost. Kot alfa in beta je program zato namesto iz vnašanja števil bral iz koordinatske pozicije računalniškega kazalca na strani.

function draw2(lenght){

    line(0,0,0,-lenght);

    translate(0,-lenght);

    if(lenght>9)

    {

        push();

        rotate((mouseX/10)/(TWO\_PI\*8));

        draw2(lenght \* 0.66);

        pop();

        push();

        rotate(-(mouseX/10\*0.3)/(TWO\_PI\*8));

        draw2(lenght \* 0.66);

        pop();

    }

}

### KOCHOVA RAVNINA

Kot drugega fraktala sem se lotil Kochove ravnine. Po deklaraciji platna v funkciji setup(), sem nato v funkciji draw() deklariral dva vektorja, in jima določil izhodiščni koordinati.

let a = createVector(0,height/1.5);

let b = createVector(width,height/1.5);

Za deklaracijo vektorjev program bere uporabnikov vnos kota obračanja in pa števila generacij ravnine.

koch\_kot = document.getElementById("kotobracanja").value;

console.log(koch\_kot);

stgeneracij = document.getElementById("stgeneracij").value;

console.log(stgeneracij);

Ker se računska zahtevnost tega fraktala z vsako generacijo večja za 4krat, se lahko to hitro pozna pri hitrosti izrisa ravnine. Zaradi tega razloga sem za branjem vrednosti vstavil if stavek, ki v primeru da je vpisano število generacij večje od 5 pokaže okensko opozorilo na strani, in vrednost števila generacij ponastavi na vrednost, ki je bila vpisana pred to.

if(stgeneracij > 5){

    stgeneracij = stgeneracijStaro;

document.getElementById("stgeneracij").value = stgeneracijStaro;

     window.alert('Izbrali ste število izven mej!')

    }

Po deklaraciji vektorjev sem iz njiju deklariral nov objekt iz razreda del, v katerem sem računal koordinate posameznih delov ravnine in pa izrisoval te dele, nato pa ta objekt shranil v tabelo delov z funkcijo .push().

let v = new del(a,b);

deli.push(v);

Nato gre program v for zanko, ki se izvede le tolikokrat, kot je določeno število generacij v programu. V zanki se nato najprej deklarira tabela “naslednja”, ki bo prenašala v vse koordinate v tabelo “deli”, in je v zanki deklarirana, ker je potrebno da je za izris vsake generacije prazna, saj se le nekatere koordinate različnih generacij ujemajo. Zanka gre nato še v eno zanko, ki pa se izvaja toliko, kolikor je dolžina tabele “deli” (vsak indeks v tabeli “deli” predstavlja eno črto). V tej zanki se nato najprej z funkcijo “računanje” izračuna razdelitev črte in se shrani v tabelo “prva”, nato pa se te koordinate z funkcijo “dodaj” le dodajo v tabelo “naslednja”, ki ima po izvedbi notranje for zanke shranjene vse koordinate za trenutno število generacij, in se nato v celoti prenese v tabelo deli, ki bo po končani zunanji zanki izrisana.

for(let x = 0; x<stgeneracij; x++){

let naslednja = [];

     for(let y = 0; y<deli.length; y++){

     prva = deli[y].racunanje();

         dodaj(prva,naslednja);

     }

     deli = naslednja;

 }

Že prej omenjena funkcija “racunanje” torej za vsak indeks v tabeli “deli” izračuna koordinate ali vektorje za naslednjo generacijo. Vsak izmed štirih delov, ki so izdelani v tej metodi, se shrani v tabelo deli, kot nov objekt v tabelo “deli”. Funkcija najprej izračuna dolžino začetne črte tako, da z funkcijo p5.Vector.sub() odšteje toćko a od točke b, nato pa jo za nadaljnje računenje deli z tri, z funkcijo .div(). Nato z funkcijo .add sešteje začetno točko a in dolžino, in ustvari vektor “prvi\_del”, ki bo predstavljal konec prve črte. V tabelo koordinate nato pod ničti indeks zapiše nov objekt (črto/vektor), z začetkom v točki a in koncem v točki “prvi\_del”. Podobno naredi tudi za četrti del, za drugi intretji del, ki pa sta rotirana, pa le ustvari vektor “drugi\_del”, ki s pomočjo vnesene vrednosti za rotacijo kota uporabnika zarotira smer vektorja. Pred vrednostjo kota je minus, ker je mreža v programu obrnjena na glavo. Funkcija po izračunu vseh koordinat le še vrne tabelo “koordinate”.

racunanje(){

let koordinate = [];

    angleMode(DEGREES);

     let dolzina = p5.Vector.sub(this.tocka\_b,this.tocka\_a);

     dolzina.div(3);

     let prvi\_del = p5.Vector.add(this.tocka\_a,dolzina);

     koordinate [0] = new del(this.tocka\_a, prvi\_del);

     let tretji\_del = p5.Vector.sub(this.tocka\_b,dolzina);

    koordinate [3] = new del(tretji\_del, this.tocka\_b);

     dolzina.rotate(-koch\_kot);

    let drugi\_del = p5.Vector.add(prvi\_del, dolzina);

     koordinate [1] = new del(prvi\_del, drugi\_del);

     koordinate [2] = new del(drugi\_del, tretji\_del);

        return koordinate;

    }

Po tem ko imamo izračunane vse vektorje, nam preostane le še izris, ki ga dosežemo z zanko, ki gre skozi vse elemente tabele “deli”, in pa funkcije “risanje”, ki bo vsak vektor posebej izrisala.

risanje(){

stroke(255);

line(this.tocka\_a.x, this.tocka\_a.y, this.tocka\_b.x, this.tocka\_b.y);

}

Ker pa hočemo, da se vsak izris avtomatko osveži, ko spremenimo katerokoli izmed vrednosti, je celotna koda za računanje zapisana v funkciji draw(), kar pomeni da se neprestano ponavlja. Zzato moramo na koncu funkcije draw() le še izprazniti tabelo “deli”, da nebi pri naslednjem izrisu narisala še prejšnjo variacijo ranvnine.

### MANDELBROTOV NIZ

Kot zadnjega, in najbolj zahtevnega fraktala za razumevanje sem se lotil Mandelbrotega niza, ki pa sem ga izdelal na nekoliko drugačen način kot ostala dva. Ker je fraktal opisan z rekurzivno funkcijo, je bilo najbolj primerno delo z piksli, kjer bo vsak piksel na skici predstavljal eno točko v koordinatnem sistemu.

Program v funkciji setup po deklaraciji platna in podaji velikosti le-tega, prvič pokliče funkcijo draw2(), ki nariše prvotni izgled fraktala, in se izvede le enkrat. Nato pa v funkciji draw() bere uporabnikov vnos vseh parametrov, ki jih je možno urejati, in kliče funkcijo draw2(), le v primeru, da je vsaj ena izmed štirih vrednosti spremenjena. Na koncu le še vse vrednosti parametrov shrani pod stare vrednosti, da jih naslednjo ponovitev lahko primerja z novimi vrednostmi, in preverja ali so spremenjene.

function draw(){

    svetilnost = document.getElementById("svetilnost").value;

    red = document.getElementById("R").value;

  green = document.getElementById("G").value;

   blue = document.getElementById("B").value;

if(redStaro != red || svetilnostStaro != svetilnost || greenStaro != green || blueStaro!= blue){

        draw2();

    }

    svetilnostStaro = svetilnost;

    redStaro = red;

    greenStaro = green;

    blueStaro = blue;

}

Funkcija draw2() torej kot že prej omenjeno opravlja računanje točk, barvanje in risanje fraktala. Funkcija najprej z funkcijo loadPixels() pošlje podatke o platnu v tabelo Pixels[], in s tem pripravi program na delo s piksli, in z funkcijo pixelDensity() nastavi njihovo gostoto na 1.

loadPixels();

pixelDensity(1);

Ker si bomo skico predstavljali kot nekakšen koordinatni sistem, v katerem bo izrisan “graf” fraktala, gre program nato v dve for zanki. Prva for zanka bo šla čez vse x koordinate osi, druga pa čez vse y koordinate grafa. Za vsako izmed točk sestavljenih iz teh x in y koordinat bomo v nadaljenavnju z pomočjo funkcije preverjali, ali spada v notranjost fraktala, ali zdrsne v neskončnost, in predstavlja zunanjost fraktala.

# ZAKLJUČEK

# VIRI IN LITERATURA

# STVARNO KAZALO

# PRILOGE