**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I**

**INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**VIZUALIZACIJA PODATAKA**

**PROJEKTNI ZADATAK**

**David Turkalj, DRC**

**Osijek, 2021.**

Sadržaj

[1. UVOD 1](#_Toc75635349)

[2. PROGRAMSKO RJEŠENJE I KORIŠTENE TEHNOLOGIJE 2](#_Toc75635350)

[2.1. HTML i CSS 2](#_Toc75635351)

[2.2. JAVASCRIPT 4](#_Toc75635352)

[3. PREPREKE I PROBLEMI TIJEKOM IZRADE 10](#_Toc75635353)

[4. ZAKLJUČAK I MOGUĆA POBOLJŠANJA 11](#_Toc75635354)

# UVOD

Tema ovog projektnog zadatka je napraviti nešto drugačije. Najveći izazov je bio smisliti temu koja već nije napravljena. Tako sam došao na ideju diobe krugova. Kako bi projekt imao smisla, morao je riješiti nekakav problem, taj problem je kreativno predviđanje oblika i boja. Vizualizira se svaka dioba kruga nakon što se pritisne. Najveći izazov je bio napraviti pamćenje i reprodukciju pritisaka kako bi se mogao vidjeti postupak izrade vizualizacije.

# PROGRAMSKO RJEŠENJE I KORIŠTENE TEHNOLOGIJE

Za izradu projekta korišteno je više različitih tehnologija koje su usko povezane. Kao osnova korišten je HTML uz JavaScript i CSS za uređivanje i raspored elemenata na stranici. Kako bi se sve potrebne animacije uredno odvijale i bez poteškoća korištena je eksterna biblioteka, Data Driven Document ili skraćeno D3.js. D3 je JavaScript biblioteka koja pomoću svojim moćnih i mnogobrojnih funkcija omogućava jednostavnu i brzu manipulaciju HTML elemenata.

## HTML i CSS

Kao korijenska tehnologija korišten je HTML u kombinaciji sa CSS-om. HTML nije programski jezik nego je skup propisa kako stranica treba izgledati, kako Internet preglednik treba renderirati stranicu. U nastavku je HTML dio ovog projekta.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <link rel="stylesheet" href="style.css">

    <title>Dots</title>

</head>

<body>

    <script src="https://d3js.org/d3.v6.min.js"></script>

    <div id="color-picker">

        <ul>

            <li>

                <label for="colorQ">Bind color to key 'Q':</label>

                <input type="color" id="colorQ" name="colorQ" value="#FF0000">

            </li>

            <li>

                <label for="colorW">Bind color to key 'W':</label>

                <input type="color" id="colorW" name="colorW" value="#00FF00">

            </li>

            <li>

                <label for="colorE">Bind color to key 'E':</label>

                <input type="color" id="colorE" name="colorE" value="#0000FF">

            </li>

            <li>

                <label for="colorR">Bind color to key 'R':</label>

                <input type="color" id="colorR" name="colorR" value="#FFFFFF">

            </li>

            <li>

                <label for="colorT">Bind color to key 'T':</label>

                <input type="color" id="colorT" name="colorT" value="#000000">

            </li>

            <li>

                <label for="backgroundColor">Background color: </label>

                <input type="color" id="backgroundColor" name="backgroundColor"

                        onchange="backgroundColorChange()" value="#D3D3D3">

            </li>

        </ul>

        <button id="replayButton" type="button" onclick="replay()">Replay</button>

        <label for="selectedColor">Current selected color:</label>

        <input disabled type="color" id="selectedColor"

                name="selectedColor" value="#FFFFFF">

    </div>

    <div id="dots">

    </div>

    <script src="dots.js"></script>

</body>

</html>

Nakon HTML dolazi CSS dio koji će taj HTML urediti i posložiti elemente da stoje i izgledaju kako želimo. CSS je jezik koji služi za manipuliranje kako će se HTML renderirati. Jako je moćan iz razloga što može promijeniti bilo koje svojstvo bilo kojeg HTML elementa, od boje, fonta teksta, pozicije na stranici pa čak i treba li se renderirati ili ne. CSS dio projekta:

html, body {

    margin: 0;

    height: 100%;

    width: 100%;

    font-size: 14px;

    font-family: "Helvetica Neue", Helvetica, Arial, Sans-serif;

    font-weight: 300;

    text-align: center;

    cursor: default;

    background-color: lightgray;

    overflow: hidden;

}

div {

    cursor: inherit;

}

#dots {

    position: relative;

    top: 50px;

    height: 100%;

    width: 512px;

    margin: 0 auto;

}

#color-picker {

    position: relative;

    text-align: left;

    margin: 0 auto;

}

p {

    font-weight: bold;

}

input {

    margin: 4px;

}

button {

    margin-left: 16px;

}

## JAVASCRIPT

U HTML dijelu se mogu vidjeti dvije linije koda u kojima se spominje „script“. Ta linija omogućuje učitavanje JavaScript koda iz druge lokalne ili online datoteke u HTML dokument. Lokalne se najčešće koriste za pojednostavljenje izgleda sveukupnog programskog koda radi čitljivosti i održavanja, dok se online koriste kako bi se na stranici implementirala neka biblioteka potrebnih funkcija. U ovom projektu koriste se obje, prva je za D3.js biblioteku dok je druga za lokalnu datoteku u kojoj se nalazi sva logika. Pozicija na kojoj se skripte učitavaju je jako bitna, u ovome slučaju da se lokalna skripta stavila odmah ispod D3-a, stranica ne radila jer skripta ne bi bila u stanju pronaći elemente koji su definirani u HTML dijelu.

Do sada više puta spomenula D3.js biblioteka, sada će se malo detaljnije opisati. D3 je kratica za Data Driven Document. To je biblioteka koja se primarno koristi za potrebe vizualizacije i animacije na web stranicama. To je moguće zbog jednostavne manipulacije HTML-a, CSS-a korištenjem JavaScript-a. Primarno kako D3 funkcionira je kreiranjem SVG elementa u kojem onda pomoću velikog broja funkcija, koje su nastale kao rezultat razvoja različitih alata, mogu manipulirati razni HTML elementi, najčešći su najjednostavniji, krug, kvadrat i malo kompleksniji „path“. Trenutna verzija D3.js biblioteke se može pronaći na [poveznici](https://github.com/d3/d3/blob/main/API.md), gdje je popis svih trenutnih funkcija i njihovo objašnjenje i primjer implementacije.

JavaScript dio projekta:

let width = 512;

let height = 512;

let currentColor = "#FFFFFF";

let duration = 1000;

let constraints = {

    startX: 0,

    startY: 0,

    endX: 0,

    endY: 0

}

let mouseClickPositions = [];

let isReplay = false;

let inter;

setConstraints();

let selectedColor = d3.select("#selectedColor", currentColor);

let body = d3.select("body")

    .on("keydown", function (event) {

        switch (event.keyCode) {

            case 81: switchColor(d3.select("#colorQ").property("value")); //Q

                break;

            case 87: switchColor(d3.select("#colorW").property("value")); //W

                break;

            case 69: switchColor(d3.select("#colorE").property("value")); //E

                break;

            case 82: switchColor(d3.select("#colorR").property("value")); //R

                break;

            case 84: switchColor(d3.select("#colorT").property("value")); //T

                break;

            default: switchColor("#FFFFFF");

                break;

        }

    });

let svg = d3.select("#dots")

    .append("svg")

    .attr("width", width)

    .attr("height", height);

let firstCircle = svg.append("circle")

    .attr("cx", width / 2)

    .attr("cy", height / 2)

    .attr("fill-opacity", 1)

    .attr("fill", "#FFFFFF")

    .attr("stroke", "#000000")

    .on("click", function (d) {

        addMouseClickPosition(event);

        split(d3.select(this));

    })

    .attr("r", 0)

    .transition()

    .duration(duration)

    .attr("r", width / 2);

function split(circle) {

    let details = {

        cx: parseInt(circle.attr("cx")),

        cy: parseInt(circle.attr("cy")),

        r: parseInt(circle.attr("r")),

        color: circle.attr("fill")

    }

    circle.transition()

        .duration(duration)

        .attr("r", 0)

        .style("opacity", 0)

        .remove();

    delete circle;

    for (let i = 0; i < 4; i++) {

        svg.append("circle")

            .attr("cx", details.cx - (details.r / 2) \* ((i == 0 || i == 1) ? 1 : -1))

            .attr("cy", details.cy - (details.r / 2) \* ((i == 0 || i == 2) ? 1 : -1))

            .attr("fill-opacity", 1)

            .attr("fill", currentColor)

            .on("click", function (d) {

                if (checkSize(details.r)) {

                    addMouseClickPosition(event);

                    split(d3.select(this));

                }

            })

            .attr("r", 0)

            .transition()

            .duration(duration)

            .attr("r", details.r / 2);

    }

}

function checkSize(r) {

    return r / 2 > width / 256;

}

function backgroundColorChange() {

    body.style("background-color", d3.select("#backgroundColor").property("value"));

}

function switchColor(color) {

    currentColor = color;

    d3.select("#selectedColor").property("value", currentColor);

}

function setConstraints() {

    constraints.startX = getStartingPosition(document.getElementById("dots")).x;

    constraints.startY = getStartingPosition(document.getElementById("dots")).y + 30;

    constraints.endX = constraints.startX + 512;

    constraints.endY = constraints.startY + 512 + 30;

}

function addMouseClickPosition(data) {

    if (!isReplay) {

        let position = {

            x: data.clientX - constraints.startX,

            y: data.clientY - constraints.startY,

            color: currentColor

        }

        mouseClickPositions.push(position);

    }

}

function getStartingPosition(element) {

    for (var lx = 0, ly = 0;

        element != null;

        lx += element.offsetLeft,

        ly += element.offsetTop,

        element = element.offsetParent);

    return {

        x: lx,

        y: ly

    };

}

function replay() {

    isReplay = true;

    document.getElementById("replayButton").disabled = true;

    d3.selectAll("circle").transition()

        .duration(duration)

        .attr("r", 0)

        .style("opacity", 0)

        .remove();

    svg.append("circle")

        .attr("cx", width / 2)

        .attr("cy", height / 2)

        .attr("fill-opacity", 1)

        .attr("fill", "#FFFFFF")

        .attr("stroke", "#000000")

        .on("click", function (d) {

            split(d3.select(this));

        })

        .attr("r", 0)

        .transition()

        .duration(duration)

        .attr("r", width / 2)

        .on("end", function () {

            inter = setInterval(playReplay, 1500)

        });

}

function playReplay() {

    if (isReplay && mouseClickPositions.length > 0) {

        let svgPosition = getStartingPosition(document.getElementById("dots"));

        let x = svgPosition.x + mouseClickPositions[0].x;

        let y = svgPosition.y + mouseClickPositions[0].y;

        let el = document.elementFromPoint(x, y);

        let ev = new MouseEvent("click",

            {

                clientX: x,

                clientY: y,

                bubbles: true

            });

        currentColor = mouseClickPositions[0].color;

        el.dispatchEvent(ev);

        mouseClickPositions.shift();

    }

    if (mouseClickPositions.length == 0) {

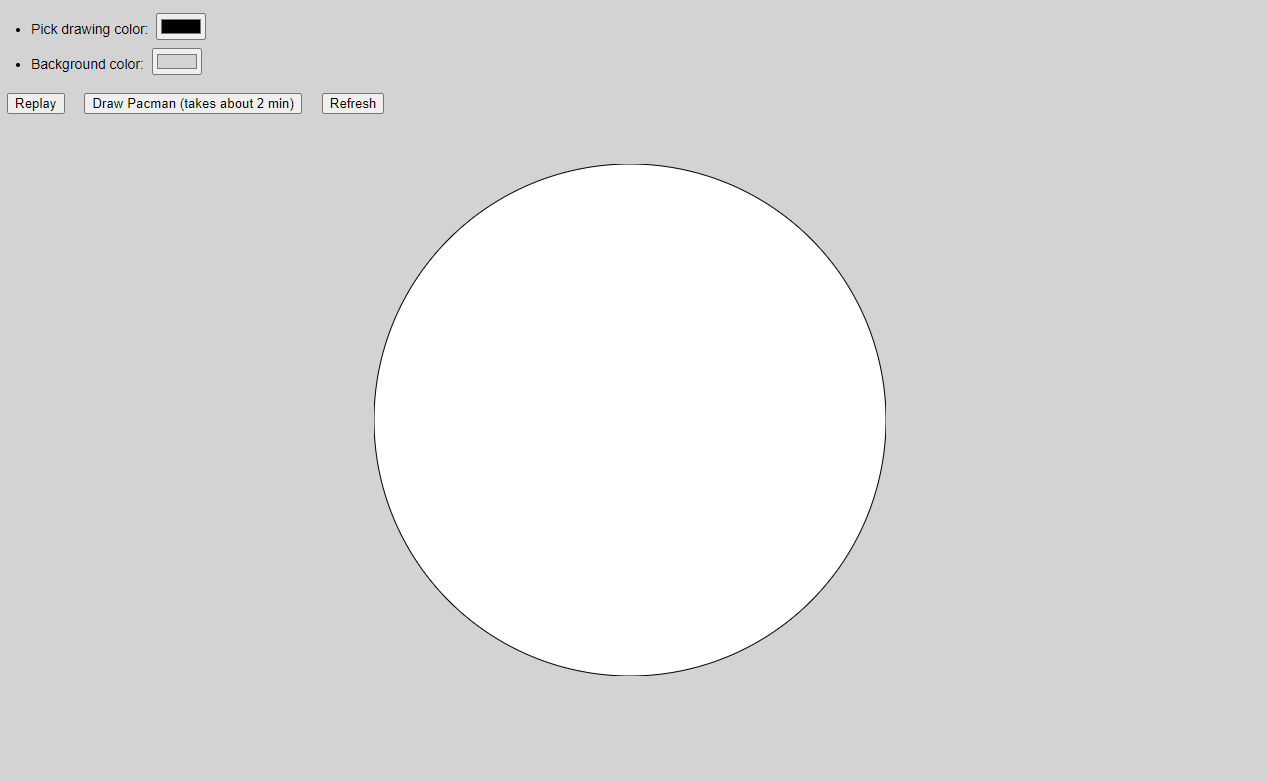
        clearInterval(inter);

        document.getElementById("replayButton").disabled = false;

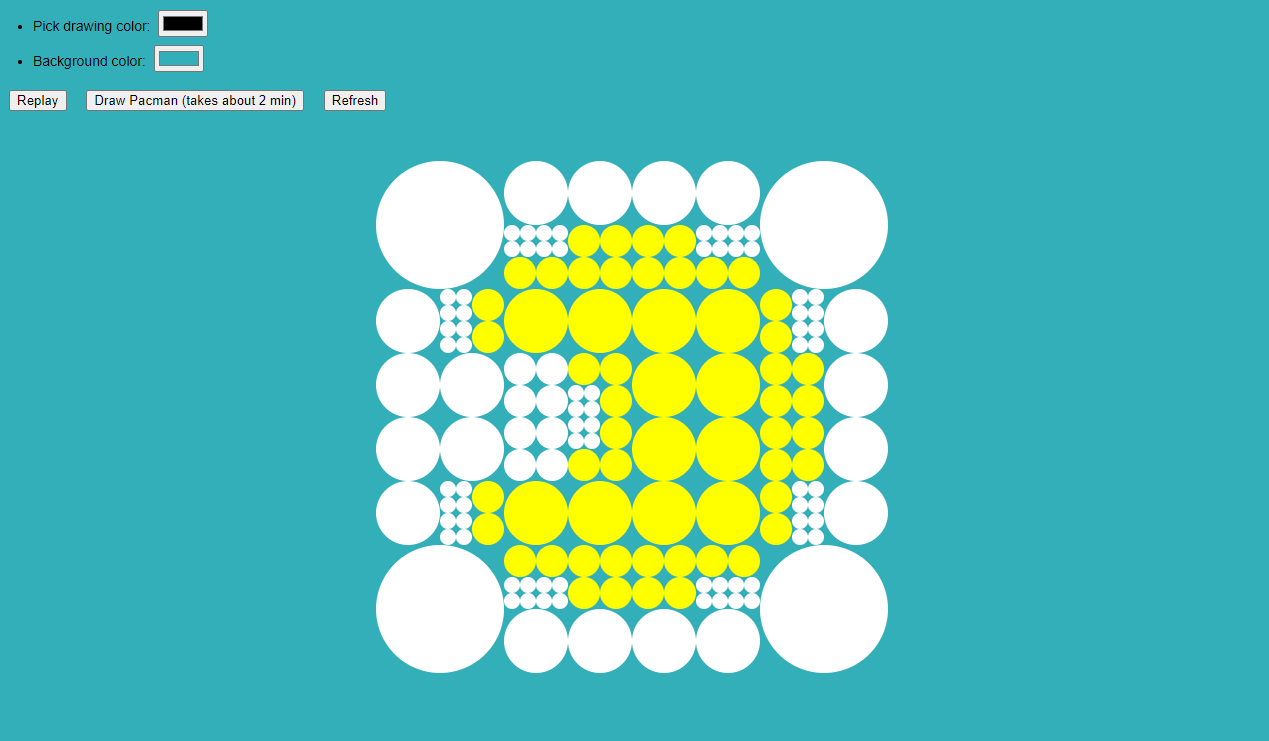
    }

}

Sada kada su sve tehnologije koje se koriste opisane i priložen programski kod može se pokazati kako stranica izgleda pri učitavanju.



Slika 1 Izgled stranice nakon što se učita



Slika 2 Izgled nakon promjene pozadine i odvijanja funkcije za crtanje

# PREPREKE I PROBLEMI TIJEKOM IZRADE

Kao i svaki projekt u početku ideja nije bila definirana do kraja i zbog toga ja prolazila kroz više verzija. Svaka iduća verzija je riješila nekakav problem iz prethodne, ali napravila novi ili jednostavno nije riješila sve probleme.

Prva verzija je napravljena tako da su boje u koje se krug dijelio bile dodijeljene na tipke na tipkovnici što je bio ogroman problem kada se stranica otvori na mobilnom uređaju. Nije bilo načina da se otvori tipkovnica, a i takav pristup nije dobar.

Iduća verzija rješava taj problem tako da se koristi samo jedan izbornik boje i koja se boja nalazi tako je ona u koju će se dioba odvijati, malo usporava postupak. Isto tako u ovoj verziji dodana je tipka „Refresh“ koja vraća stanje cijele stranice na početno.

U idućoj verziji je došao najteži izazov do sada. Implementirati povijest klikova i napraviti mogućnost reprize. Tu je najviše pomoglo što je prešlo za verzije 3 D3.js na verziju 6, koja ima nove funkcije koje su jako korisne. Najveća korist je bila funkcija on(„end“) koja se odvija nakon što se prethodni zadatak odradi.

Tu je razvoj stao na neko vrijeme i mislio sam da sam gotov, ali nešto je nedostajalo, nešto što može demonstrirati što se može napraviti i koliko je planiranja potrebno da se nešto tako odradi. Implementirao sam funkcionalnost crtanja poznatog lika iz video igre. I tu se može vidjeti koliko je nešto tako jednostavno zapravo kompleksno za izvesti jer je potrebno razmišljati nekoliko koraka unaprijed. Isto tako ostao mi je jeda problem koji ne znam kako riješiti, a to je blokiranje korisnika da pritisne krug dok animacija traje.

Poveznica na Github gdje se nalazi svaka verzija: <https://github.com/Folen0909/VIZ_Dot_drawing>

Poveznica na stranicu: <http://viz-dot-drawing.herokuapp.com/>

# ZAKLJUČAK I MOGUĆA POBOLJŠANJA

Ideja projekta je zanimljiva i kako je implementirana potiče korisnika da razmišlja više koraka unaprijed kako će završna vizualizacija izgledati, dok se u isto vrijeme odvikava linearnog razmišljanja piksel crtanja. Daljnje funkcionalnosti koje bi se mogle implementirati su spremanje izrađenog crteža te dodavanje spremanje sekvencije klikova i učitavanje.