Vizualizacija podataka

LV3

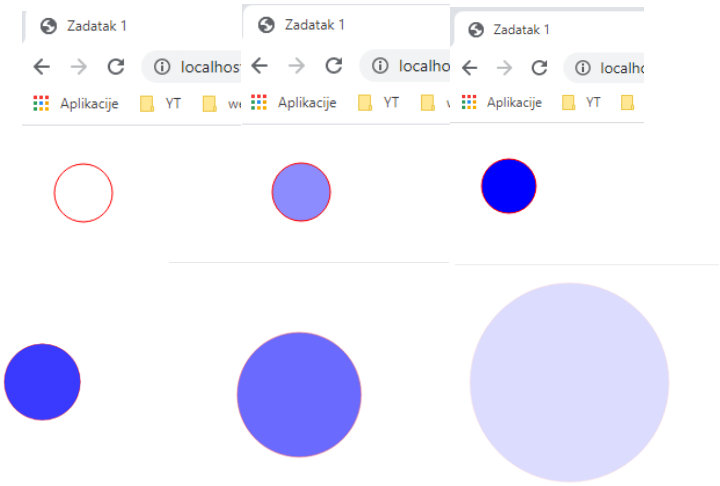
ANIMACIJA I PRIDRUZIVANJE PODATAKA

Student: Kristijan Koščak

Smjer: DRC, 1.godina

Datum: 10.04.2020.

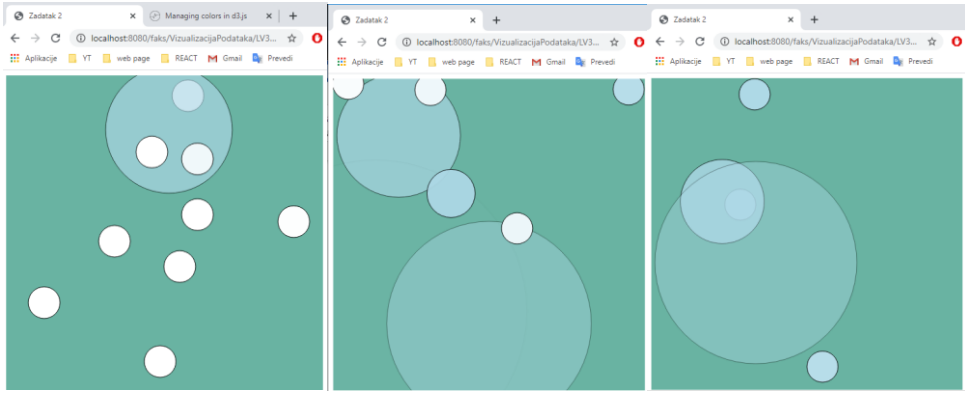
Zadatak 1:




```
var svg = d3.select("body")
    .append("svg")
    .attr("width", 400)
    .attr("height", 400)

var rects = svg.append("circle")
    .attr("cx", 200)
    .attr("cy", 200)
    .attr("r", 25)
    .style("stroke", "red")
    .style("fill", "none");

d3.select("circle")
    .style("fill", "white")
    .transition()
    .delay(3000)
    .duration(5000)
    .style("fill", "blue")
    .transition()
    .duration(5000)
    .ease("linear")
    .attr("r", svg.attr("width")/2)
    .style("opacity",0);
```

Na prve tri slike vidljiva je ispuna našeg kruga kao što je i zahtijevano. Na druge tri slike vidimo povećavanje kruga na vrijednost svgWidth/2 i smanjivanje atributa *opacity* na 0. Kako bi sve glatko tekla tranzicija boja, postavili smo boju prije tranzicije u bijelu. Ukoliko se ne postavi, tranzicija kreće iz crne prema boji ispune koju smo postavili jer je prema *default*-u boja ispune crna. Samim time promjena boja mora imat nekakvu početnu i krajnju vrijednost(boju) stoga smo taj problem riješili na ovaj način.

Zadatak 2:

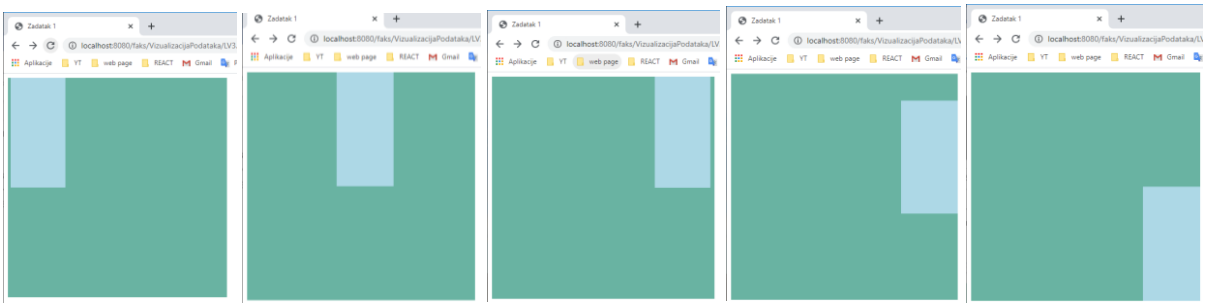


```
var svg = d3.select("body")
    .append("svg")
    .attr("width", 500)
    .attr("height", 500)
    .style("background-color", "#69b3a2");

function makeCircles() {
    var circles = 10;
    var data = [];
    for (var i = 0; i < 10; i++) {
        data.push(i);
    }
    svg.selectAll("circle")
        .data(data)
        .enter()
        .append("circle")
        .attr("cx", function (d) {
            return Math.round(Math.random() * (500 - 25))
        })
        .attr("cy", function (d) {
            return Math.round(Math.random() * (500 - 25))
        })
        .attr("r", 25)
        .attr("fill", "white")
        .style("stroke", "black")
        .transition()
        .delay(function (d) {
            return Math.round(Math.random() * 2500)
        })
        .duration(function (d) {
            return Math.round(Math.random() * 5000)
        })
        .attr("fill", "lightblue")
```

```
            .transition()
            .duration(function (d) {
                return Math.round(Math.random() * 3000)
            })
            .ease("linear")
            .attr("r", svg.attr("width") / 2)
            .style("opacity", 0)
            .each("end", function () {
                d3.select(this).remove();
                circles--;
                if (circles == 0) {
                    makeCircles();
                }
            });
    }
    makeCircles();
```

Za razliku od prethodnog zadatka, ovdje smo automatizirali dodavanje 10 kružnica s različitim vrijednostima trajanja tranzicije i položajima(x i y). Koristeći se Math.random() * X , gdje X predstavlja maksimalnu vrijednost dobili smo nasumične brojeve do vrijednosti X. Većina koda je ista kao i u prethodnom zadatku uz male preinake zbog automatizacije. Nakon što se na svakoj kružnici obavi zadnja tranzicija, brišemo ju i umanjujemo brojač kružnica. Kad on dođe na 0 tj. kad više nema kružnica, vršimo rekurzivni poziv funkcije koja obavlja sve gore navedeno.

Zadatak 3:

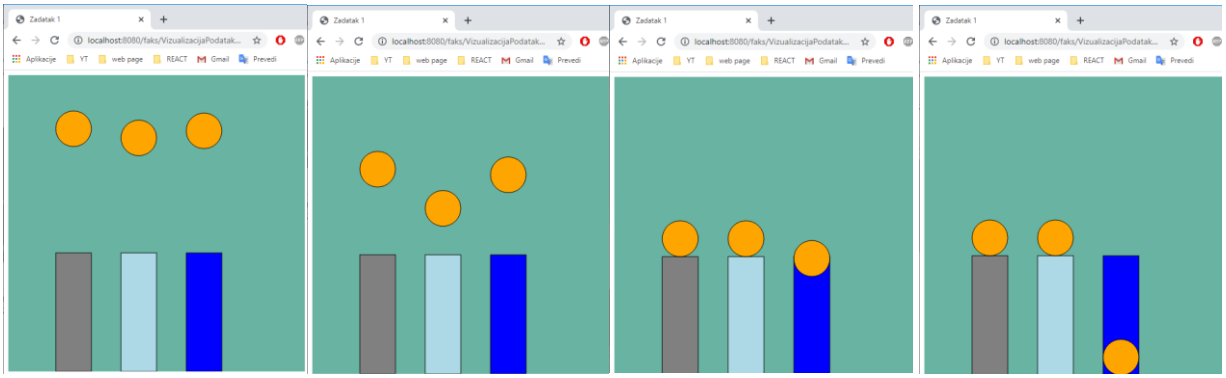


```
    var svg = d3.select("body")
        .append("svg")
        .attr("width", 400)
        .attr("height", 400)
        .style("background-color", "#69b3a2");

    var rects = svg.append("rect")
        .attr("x", 0)
        .attr("y", 0)
        .attr("width", 100)
        .attr("height", 200)
```

```
            .style("fill", "lightblue");

        d3.select("rect")
            .transition()
            .delay(250)
            .duration(1000)
            .attr("transform", function (d) { return "translate(" + (svg.attr("width") - rects
.attr("width")) + ")"; })
            .transition()
            .duration(1000)
            .attr("transform", function (d) { return "translate("+(svg.attr("width") - rects.a
ttr("width"))+","+ (svg.attr("height") - rects.attr("height")) + ")"; });
```

Dodali smo pravokutnik u položaj 0,0 kao što je i zahtijevano te smo ga sredili. Nakon toga smo napravili dvije tranzicije gdje smo ga pomoću transformacije , translatirali prvo po X osi, a zatim po Y na zadane vrijednosti. Trajanje svake tranzicije je 1s.

Zadatak 4:



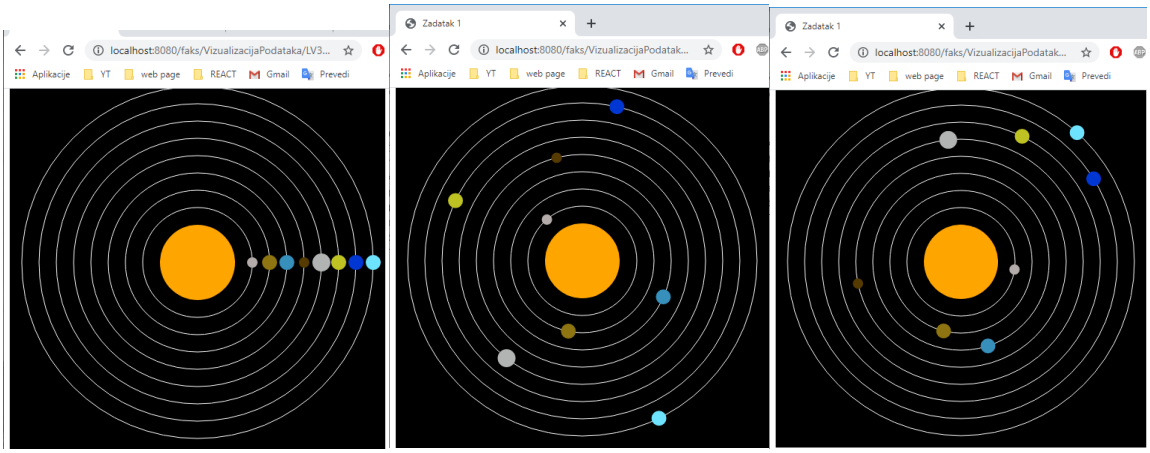
```
var svg = d3.select("body")
            .append("svg")
            .attr("width", 500)
            .attr("height", 500)
            .style("background-color", "#69b3a2");

        var distance = [110, 220, 330];
        var colors = ["gray", "lightblue", "blue"];

        svg.selectAll("rect")
            .data(distance)
            .enter()
            .append("rect")
            .attr("x", function (d, i) {
                return 0 + distance[i] - 30;
            })
```

```
            .attr("y", svg.attr("height") - 200)
            .attr("width", 60)
            .attr("height", 200)
            .attr("fill", function (d, i) { return colors[i] })
            .style("stroke", "black");

        svg.selectAll("circle")
            .data(distance)
            .enter()
            .append("circle")
            .attr("cx", function (d, i) {
                return 0 + distance[i];
            })
            .attr("cy", 50)
            .attr("r", 30)
            .attr("fill", "orange")
            .style("stroke", "black");


        var circles = d3.selectAll("circle");
        d3.select(circles[0][0])
            .transition()
            .duration(1500)
            .ease("linear")
            .attr("transform", function (d) {
                return "translate(0,220)";
            });
        d3.select(circles[0][1])
            .transition()
            .duration(1500)
            .ease("bounce")
            .attr("transform", function (d) {
                return "translate(0,220)";
            });
        d3.select(circles[0][2])
            .transition()
            .duration(1500)
            .ease("linear")
            .attr("transform", function (d) {
                return "translate(0,240)";
            })
            .transition()
            .delay(1500)
            .duration(2500)
            .attr("transform", function (d) {
```

```
            return "translate(0,420)";
        });
```

Na prve četiri slike vidimo simulaciju slobodnog pada kugli i reakciju pri udaru od pravokutnike koji redom predstavljaju krutu tvar, elastičnu tvar i tekućinu. Padom kugle na čvrstu tvar, ona se zadržava na površini te tvari. Kod elastične tvari, kugla odskakuje neko vrijeme te se u konačnici zadržava na površini. Kod tekućine, kugla pri udaru od površinu malo zaranja, usporava te tone ka dnu. Na slikama je prikazan pad sve tri kugle u periodu od 1,5 sekunde. Postoje male razlike između tri kugle. Kod prve kugle, navedena simulacija se događa u periodu od 1,5 sekunde odnosno pad i zadržavanje na površini. Kod druge, udar i odskakivanje se odvija unutar 1,5 sekunde te ona brže dotakne površinu od prve i treće kugle. Kako bi sve bilo jednako, potrebno je podesiti period trajanja pada druge kugle na 3 sekunde kako bi uočili razliku. Kod treće kugle imamo dva dijela simulacije, udar od površinu i usporavanje te padanje ka dnu.

Zadatak 5:



```
    var planets = [
        {
            "color": "orange",
            "size": "50",
            "rotation": 0,
            "distance": 0,
            "speed": 0
        },
        {
            "color": "#B3ACAA",
            "size": "7",
            "rotation": 0,
            "distance": 50,
            "speed": 11 * Math.PI / 180
```

```
        },
        {
            "color": "#8E7512 ",
            "size": "10",
            "rotation": 0,
            "distance": 50,
            "speed": 8 * Math.PI / 180
        },
        {
            "color": "#378FBB",
            "size": "10",
            "rotation": 0,
            "distance": 50,
            "speed": 9 * Math.PI / 180
        },
        {
            "color": "#553B00  ",
            "size": "7",
            "rotation": 0,
            "distance": 50,
            "speed": 6 * Math.PI / 180
        },
        {
            "color": "#B2B3B3",
            "size": "12",
            "rotation": 0,
            "distance": 50,
            "speed": 3 * Math.PI / 180
        },
        {
            "color": "#BEC223",
            "size": "10",
            "rotation": 0,
            "distance": 50,
            "speed": 2 * Math.PI / 180
        },
        {
            "color": "#0037D2",
            "size": "10",
            "rotation": 0,
            "distance": 50,
            "speed": 1 * Math.PI / 180
        },
        {
            "color": "#6EE3FF",
```

```
            "size": "10",
            "rotation": 0,
            "distance": 50,
            "speed": 1.5 * Math.PI / 180
        }
    ];

    var distance = [];
    for (var i = 0; i < 9; i++) {
        planets[i].distance += (i * 23);
    }
    var svg = d3.select("body")
        .append("svg")
        .attr("width", 500)
        .attr("height", 500)
        .style("background-color", "black");


    svg.selectAll("rect")
        .data(planets)
        .enter()
        .append("circle")
        .attr("cx", svg.attr("width") / 2)
        .attr("cy", svg.attr("height") / 2)
        .attr("r", function (d) { return d.distance; })
        .attr("fill", "none")
        .attr("stroke", "white")
        .attr("stroke-width", 1);

    svg.selectAll("rect")
        .data(planets)
        .enter()
        .append("circle")
        .attr("cx", function(d){return svg.attr("width") / 2 +d.distance ;})
        .attr("cy", svg.attr("height") / 2)
        .attr("r", function (d) { return d.size })
        .attr("fill", function (d) { return d.color })


    setInterval(update, 100);

    function update() {

        var circles = d3.selectAll("circle");
        for (var i = 10; i < 18; i++) {
```

```
                d3.select(circles[0][i])
                    .transition()
                    .delay(0)
                    .duration(100)

                    .attr('cx', function (d) { return svg.attr("width") / 2 + Mat
h.cos(d.rotation) * d.distance; })
                    .attr('cy', function (d) { return svg.attr("height") / 2 + Ma
th.sin(d.rotation) * d.distance; });
            }
            for (var planet in planets){
                planets[planet].rotation += planets[planet].speed;
            }
        }
```

Radi lakšeg rukovanja podacima koristili smo znanje stečeno na prethodnim vježbama, a to je JSON format podataka. Ključni parametri su nam *rotation*, *distance* i *speed* ! Prvi parametar predstavlja trenutni položaj planeta u odnosu na ishodište tj. kut, drugi je udaljenost putanje pojedinog planeta od ishodišta i treći je brzina odnosno kut za koji se planet pomakne svakim korakom. Prvi zadatak nam je bio iscrtati putanje planeta koje nam predstavljaju kružnice bez ispune. Svaka ima bijeli obrub debljine 1 i polumjer određenog iznosa. Nakon toga smo iscrtali planete (kružnice s ispunom) koje smo trebali smjestiti na odgovarajuće putanje. Razmještali smo ih prema x osi za iznos zbroja ishodišta i *distance.* Bitna stavka ovoga zadatka je metoda setInterval kojom svakih 0.1 sekundi pozivamo funkciju kojom omogućavamo kretanje planeta po putanjama. To smo uradili tako što smo dohvatili sve kružnice i izbirali planete. Imamo sveukupno 18 kružnica(putanja i planeta). Nakon toga smo koristeći se tranzicijom postavljali vrijednost cx i cy svakog planeta. Vrijednost na koju smo ih postavljali je zbroj vrijednosti x i y od ishodišta i trigonometrijskih funkcija kosinusa i sinusa kuta za koji se pomakne(*rotation*) pomnožen sa udaljenošću svake putanje od ishodišta. Nakon toga svakom planetu povećavamo kut za iznos *speed*-a.