UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Tadej Vrtovec

Vizualizacija zvočnih zbirk

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Matija Marolt

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V nalogi preglejte različne načine vizualizacij podatkov, s poudarkom na vizualizaciji na spletu z modernimi tehnologijami HTML5 in JavaScript. Opišite nekaj tipov bolj kompleksnih podatkovnih vizualizacij ter podajte njihove prednosti in slabosti. Na podlagi zaključkov razvijte ustrezno vizualizacijo zbirke slovenskih ljudskih pesmi.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Tadej Vrtovec z vpisno številko **63010354** sem avtor diplomskega dela z naslovom:

Vizualizacija zvočnih zbirk

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matija Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela.

V Ljubljani, 29. oktobra 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
AJAX	Asynchronous JavaScript and XML	Asinhroni Javascript in XML
AMF	Action Message Format	format za akcijska sporočila
API	Application Programming Interface	aplikacijski programski vmensik
BSD	Berkeley Software Distribution	Berkeley distribucija programske opreme
CSS	Cascading Style Sheets	kaskadne stilske podloge
DOM	Document Object Model	dokumentno objektni model
GIS	Geographic Information System	geografski informacijski sistem
GPL	GNU General Public License	splošno dovoljenje GNU
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
ISA	Industry Standard Architecture	standardna arhitektura industrije
JS	JavaScript	JavaScript
JSON	JavaScript Object Notation	objektna notacija JavaScript
RIA	Rich internet application	bogate spletne aplikacije
SVG	Scalable vector graphics	umerljiva vektorska grafika
SWF	Small Web Format	majhen spletni format
TLD	Top Level Domain	vrhnja internetna domena
USB	Universal Serial Bus	univerzalno serijsko vodilo
XML	Extensible Markup Language	razširljiv označevalni jezik

Povzetek

Vizualizacija podatkov je interdisciplinarna veda, ki se ukvarja s predstavitvijo podatkov. Zadnji čas se še posebej uporablja na svetovnem spletu, v medijih in drugih raziskovalnih panogah z namenom, da bi širši javnosti na čim boljši način prikazala vzorce v zbranih podatkih. V tem diplomskem delu bomo pogledali trenutno najbolj razširjene programske vmesnike in načine za izdelavo vizualizacij na spletu. Največ pozornosti bomo posvetili knjižnici D3.js. Opisali bomo nekaj tipov kompleksnejših podatkovnih vizualizacij ter podali njihove prednosti in slabosti. Dotaknili se bomo tudi standardov in formatov za opisovanje podatkov in grafik. Ker se računalništvo razvija s čedalje večjo hitrostjo in število podatkov narašča, bo njihova vizualizacija dobila vedno večji pomen. Zato smo se odločili, da naredimo pregled področja in poizkusimo podati nekaj dobrih praks.

Ključne besede: vizualizacija podatkov, vrste podatkovne vizualizacije, d3.js, JSON, SVG, CSS, splet.

Abstract

Data visualization is an interdisciplinary science that deals with data presentation. Currently it is particularly used on the world web, in the media and other exploratory fields, with purpose of presenting patterns in data to wider audiences. In this thesis, we will focus on the most common programming interfaces and methods for building visualizations on the web. The most attention will be devoted to library D3.js. We shall describe some types of more complex data visualisations and find their strengths and weaknesses. We will also look at web standards and formats for describing data and graphics. As computer technology evolves with increasing speed and amount of data grows, we predict that visualisation will become even more important subject. That is why we have decided to make an overview of the field and try to find some good examples.

Keywords: data visualization, types of data visualization, d3.js, JSON, SVG, CSS, web

Poglavje 1 Uvod

Ljudje smo duhovna in intelektualna (razumska) bitja, ki s čutili zaznavamo svet okoli sebe. Vid je eden izmed petih osnovnih čutil, ki je izredno pomemben, saj z njim najlažje prepoznamo vzorce v naravi. To nam še posebej pomaga pri zagotavljanju varnosti, lagodja in napredka. Da bi se čim bolje prilagodili na okolico in njene dejavnike, moramo znati pravilno interpretirati in analizirati signale, ki nam jih posredujejo naša čutila. Že star pregovor pravi, da slika pove več kot tisoč besed. Povedano drugače: pomeni, da vsebuje več informacij. Določene izmed njih so lahko v danem trenutku odločilnega pomena, a jih zaradi slabe interpretacije ne znamo izluščiti. Kot primer vzemimo tigra, ki se skriva v travi. Če okolico pogledamo površno in si ne vzamemo dovolj časa, da bi jo analizirali, se lahko kaj hitro zgodi, da spregledamo ključno informacijo.

Povzamemo torej lahko, da je vizualizacija informacij, prejetih iz čutil kot tudi iz ostalih podatkov, pomembna pri napredovanju človeštva, ker lahko z njeno pomočjo lažje spoznavamo svet in naravo okoli sebe in si s tem zagotovimo boljše življenje.

Za napredek širše družbe pa je pomemben tudi prenos podatkov in dognanj. Tukaj imamo v mislih človeško vizualno izražanje. Že v pradavnini so si ljudje sporočali informacije v obliki primitivnih risb (jamske poslikave). Danes pa si lahko s sodobno tehnologijo podatke podajamo na bolj sofisticiran način in ob uporabi interakcije hitreje odkrijemo glavno sporočilo, ki ga vizualizacija nosi.

Računalniki so še posebej primerni za ta opravila, saj so zelo hitri in natančni pri avtomatiziranih obdelavah informacij. Vendar kljub tem pozitivnim lastnostim od njih nimamo koristi, če ne zmorejo kakovostno prikazati pridobljenih rezultatov. Zato potrebujemo dobro izhodno strojno opremo, ki je na srečo čedalje bolj dostopna. Danes imamo na voljo veliko naprav, od senzorjev za mehansko zaznavanje do raznovrstnih naprav za prikazovanje.

Da bi na podlagi našega angažiranja pri zbiranju podatkov imela korist širša družba, je smotrno, da jih vizualiziramo in jih ponudimo na ogled preko svetovnega spleta. To storimo z uporabo katere izmed spletnih tehnologij, po možnosti takšnih, ki sledijo standardom, ki jih

zahteva konzorcij W3C. S tem dosežemo visoko stopnjo prenosljivosti (portabilnost) in morebitno združevanje (integracijo) z drugimi aplikacijami.

Nadalje bomo pregledali sodobne rešitve za vizualizacijo podatkov v računalništvu. Še posebej pa se bomo osredotočili na orodja za njihovo izdelavo na svetovnem spletu in na popularne formate za izmenjevanje podatkov med njimi. Glede na to, da je vizualizacija podatkov relativno novo področje oz. področje, ki se še razvija, bomo poskušali podati nekaj dobrih praks. Pogledali bomo, kje moramo biti pozorni na težave in kako čim bolje implementirati rešitev. Našteli bomo tudi nekaj zanimivih tipov vizualizacij in jih opisali. Na koncu bomo podali še zgodovinski pregled vizualizacije podatkov.

Poglavje 2 Vizualizacija podatkov

Vizualizacija podatkov je formalno definirana kot moderna veja opisne statistike. Obsega izdelavo, proučevanje in upodobitev podatkov. Njen glavni cilj je predstaviti informacije v jasni in učinkoviti obliki, s tabelami in grafi. Uporabniki na ta način lažje analizirajo in prepoznajo vzorce v naboru prikazanih podatkov [1].

Podatkovne vizualizacije lahko ločimo glede na število različnih podatkovnih dimenzij, ki jih prikazujejo. S tem mislimo na diskretne podatkovne tipe, ki so vizualno kodirani v diagramu. Na primer: Imamo preprost graf, ki prikazuje kotacijo delnice. Parametra čas in vrednost predstavljata dve dimenziji. Ko na graf dodamo še kotacijo ene ali več drugih delnic, dobimo s tem tretjo dimenzijo. Če pa želimo prikazati še volumen trgovanja, pa to predstavlja četrto. Število dimenzij podaja stopnjo kompleksnosti vizualizacije, na katero pa ne vpliva večja količina prikazanih podatkov. Zaradi vseh teh razlogov pride pri oblikovanju podatkovnih vizualizacij do izzivov pri interakciji in pri iskanju ciljnih podatkov.

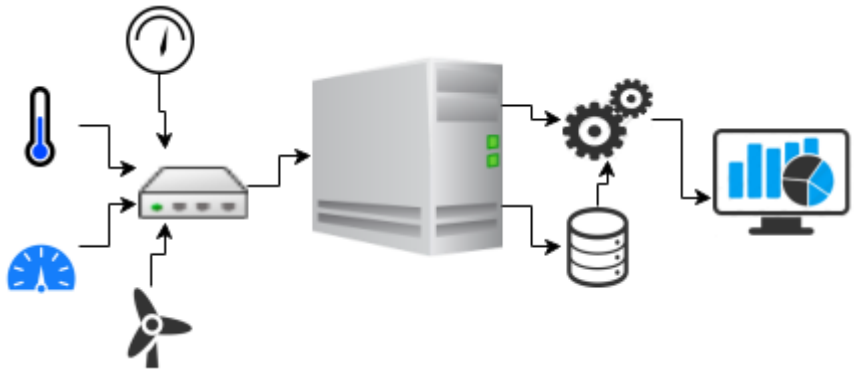
Več kot je prikazanih podatkovnih dimenzij, več unikatnih grafičnih tipov (elementov, likov, gradnikov) potrebujemo za prikaz. Pri manj kompleksnih vizualizacijah je take grafične tipe lažje izbrati ali ustvariti kot pri bolj kompleksnih. To predstavlja tudi problem pri predstavitvi (vizualizaciji) relacij med njimi. Zaradi pomanjkanja konvencij, standardov in dobrih praks predstavlja še poseben izziv združitev vseh dobrih odločitev v skupno celoto, iz katere bo lahko uporabnik pridobil želene podatke. V praksi se izkaže, da so najboljše in najbolj pogoste vizualizacije tiste, ki prikazujejo do tri podatkovne dimenzije [2].

Da bi bila vizualizacija na določenem naboru podatkov najbolje narejena, ne sme biti preveč monotona niti preveč sofisticirana. Uporabiti mora pravo mero estetike in funkcionalnosti; se pravi forma in funkcija morata biti med seboj pravilno uravnoteženi. Če se predstavitev podatkov preveč posveča grafični obliki, potem bodo uporabniki težko razbrali informacije. Zato mora idealna vizualizacija komunicirati jasno in stimulirati interakcijo, angažiranje in pozornost ter opravljati naslednje naloge:

- prikazati podatke,
- izzvati uporabnika, da bi razmišljal o vsebini in ne o metodologiji, grafičnem oblikovanju, tehnologiji grafičnega prikaza ali o čem drugem,
- izogibati se popačenju tega, kar bi morali podatki predstavljati,

- predstaviti veliko številk na majhnem prostoru,
- veliki podatkovni nabori naj bodo med seboj povezani in odvisni,
- izzvati oči, da bodo primerjale različne delčke podatkov,
- prikazati podatke na več različnih nivojih detajlov, od širokega pregleda do drobnih struktur,
- služi naj jasnemu namenu: opis, raziskovanje, dekoracije [3].

Prvi korak pri izdelavi podatkovne vizualizacije je pridobivanje podatkov. V računalništvu to poteka preko zajemanja signalov iz realnega sveta s pomočjo senzorjev. Senzorji pretvorijo fizične podatke v električne signale. Te signale se nato najprej ojači in izolira, nato pa še filtrira. Po teh obdelavah se jih s pomočjo analogno-digitalne konverzije pretvori v digitalni format. Za te namene obstaja tudi specializirana strojna oprema (data acquisition), ki je lahko preko namenskih vodil, kot so na primer serijsko vodilo, ISA, USB, PCI itd., v obliki modulov priključena na računalnik. Ko so podatki digitalizirani, se jih preko strojnih gonilnikov s pomočjo programskih sistemov shrani. Na tej točki je zajemanje končano, in lahko se prične z njihovo interpretacijo in obdelavo.



Slika 1: Diagram toka podatkov: od zajema s senzorji, digitalizacije, shranjevanja, obdelave do prikaza

Za to imamo na voljo več vrst obdelav. Prva v vrsti je podatkovna analiza, ki se ukvarja s proučevanjem podatkov. Z njo želimo izluščiti uporabne informacije in priti do zaključkov, ki nam bodo pomagali pri reševanju problema. To dosežemo s povzemanjem podatkovnih karakteristik in z uporabo vizualnih metod oz. z vizualizacijo. Druga izmed možnih obdelav je podatkovno rudarjenje, ki se ukvarja z urejanjem in pridobivanjem pomembnih informacij iz večjih količin podatkov. To dosega s pomočjo umetne inteligence, strojnega učenja, statistike in z optimizacijo podatkovnih baz. Njen namen je odkrivanje vzorcev in koristnih informacij

za nadaljnjo rabo. Podatkovna analiza in podatkovno rudarjenje sta sorodni vedi, pri katerih ima zelo pomembno vlogo podatkovna vizualizacija, saj brez nje ne moremo kakovostno razbrati rezultata obdelave.

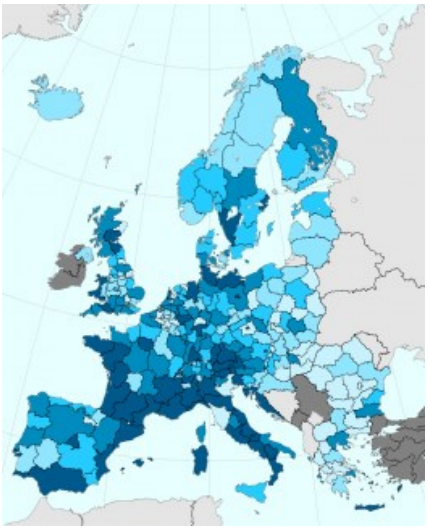
Naštejmo še nekaj ved, ki se tudi poslužujejo podatkovne vizualizacije: opisna statistika, informacijska vizualizacija, znanstvena vizualizacija, raziskovalna podatkovna analiza in statistična vizualizacija. V poslovnem svetu pa se vizualizacija uporablja predvsem v javnih ustanovah, finančnem sektorju in v vseh drugih branžah, ki se ukvarjajo z raziskavami, razvojem in analizo informacij [4]. Uporablja se tudi v računalništvu pri aplikacijah za agilni razvoj programske opreme in pri informacijskih sistemih.

2.1 Vrste vizualizacij

Ko so podatki dokončno obdelani in analizirani, je čas za njihovo predstavitev. Na tem mestu je treba postaviti robne pogoje in se na njihovi podlagi odločiti za ustrezen tip prikaza. Z robnimi pogoji imamo v mislih sledeča vprašanja: kako veliko površino imamo na voljo za prikaz, koliko in kakšno dimenzijo podatkov želimo prikazati, kakšen bo način interakcije, kdo je ciljno občinstvo, ali bo vizualizacija objavljena na spletu itd. Šele ko smo si odgovorili na vsa ta in druga zastavljena vprašanja, lahko izberemo primerno vrsto vizualizacije. Naslednji korak je definicija barvnih shem, likovnih oblik podatkov, merskih osi, morebitno dodajanje legende, umestitev teksta v vizualizacijo in ostalih stilskih elementov. Da bi uporabnika še bolj spodbudili k raziskovanju, dodamo še funkcije za interakcijo, ki omogočajo sortiranje, filtriranje, karakteriziranje itd. Na koncu pa lahko celoten prikaz še animiramo.

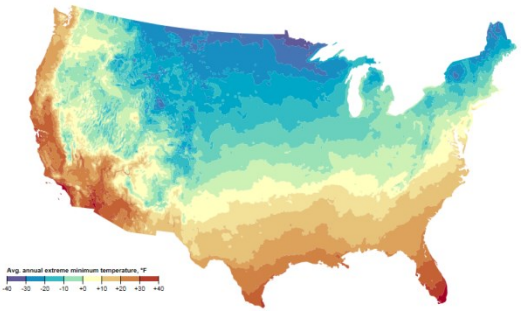
V nadaljevanju bomo pregledali nekaj primerov kompleksnejših vizualizacij podatkov. Vsak diagram bomo na kratko opisali in podali njegove prednosti. Na tem mestu lahko še povemo, da sicer obstaja še veliko drugih vrst vizualizacij, posebej tistih bolj osnovnih, s katerimi pa se je marsikdo že srečal. Nekaj jih še omenimo: tortni diagrami, histogram, stopničast diagram itd.

Choropleth so zelo priljubljeni zemljevidi za prikaz raznovrstnih podatkov. Pojavili so se že leta 1819 [5]. Na njih so regije in območja senčena z različnimi niansami barv, proporcionalno glede na zbrane meritve. Taki zemljevidi zelo učinkovito prikazujejo variiranje vzorcev na geografskem področju. Temnejša kot je barva v regiji, višja je vrednost, ki jo prikazuje. Na primer regije z več prebivalci so temnejše barve. Ta tip vizualizacije je zelo pogost pri prikazovanju statističnih podatkov volitev, brezposelnosti, osebnih dohodkov in drugih demografskih podatkov.



Slika 2: Prikaz zemljevida choropleth

Isopleth je podoben zemljevid choroplethu. Soroden mu je v tem, da prikazuje podatke na geografskem področju z različno intenziteto barv. Namesto regij pa uporablja izohipse. Z njimi doseže mehkejše (bolj gladke) prehode med različnimi nivoji. Isopleth je zelo primeren za prikaz postopnih sprememb v prostoru, manj pa za prikaz prekinjenih distribucij. Za kakovostno predstavitev potrebuje večji podatkovni nabor. Uporablja se še posebej pri merjenju temperature, padavin, zračnega tlaka in drugih vremenskih ter geoloških dejavnikov.



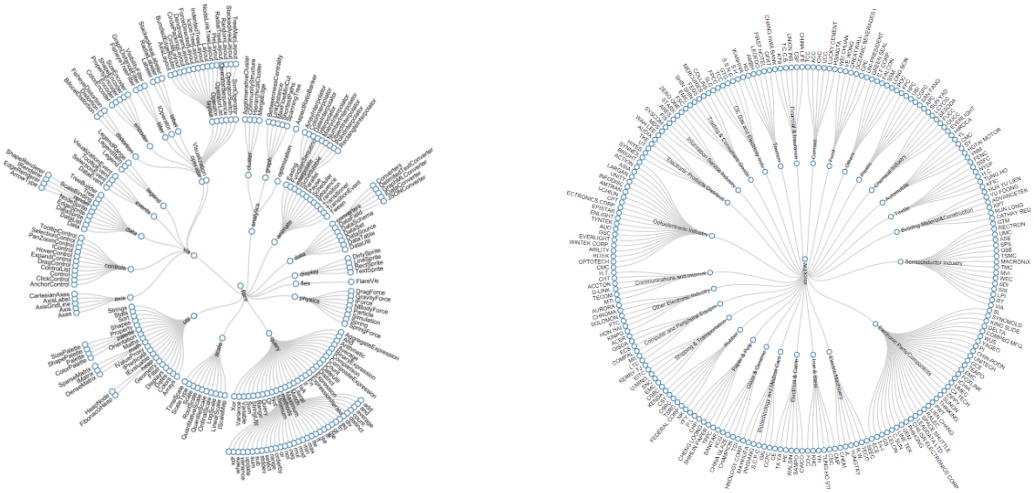
Slika 3: Isopleth

Tag Cloud ali besedni oblak je vizualna predstavitev tekstualnih podatkov, ki se po navadi uporablja za prikaz ključnih besed ali meta podatkov na spletnih straneh. Z njim želimo uporabniku čim bolj zgoščeno predstaviti vsebino in ga spodbuditi k branju. Lahko je mišljen tudi kot meni kategorij ali povezav. Značilnost besednega oblaka je, da so besede ki se bolj pogosto uporabljajo v tekstu, večje in izrazitejše barve. Prikazane so tudi bolj na sredini. Manj pogoste besede pa so manjše in prikazane bolj proti zunanjem delu. Besedni oblaki se

uporabljajo za prikazovanje trendov v spletnih iskalnikih, socialnih omrežjih in novičarskih portalih [6][7].

Slika 4: Na levem delu je klasičen besedni oblak, na desnem pa je posebna različica, ki prikazuje domenske končnice(TLD) glede na državo

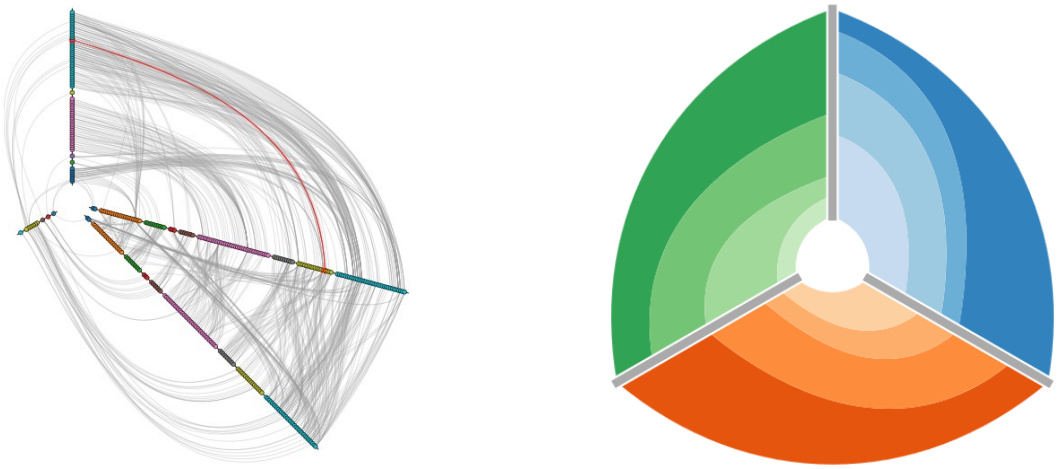
Krožni dendrogram je drevesni diagram, ki ima ciklično razporejena vozlišča. Z njim lahko zelo pregledno prikažemo hierarhijo in sorodnost elementov. To storimo najprej z uporabo funkcije za hierarhično grozdenje (clustering), ki združi elemente na podlagi njihovih podobnosti v različne grupe ali vozlišča. Nato pa jih glede na nivoje razporedimo v krog. Če drugi del izpustimo, dobimo kartezično obliko diagrama, ki pa zasede več prostora na zaslonu. Diagram se pogosto uporablja v biologiji in pri drugih panogah, ki se ukvarjajo s klasifikacijo.



Slika 5: Krožni dendrogram

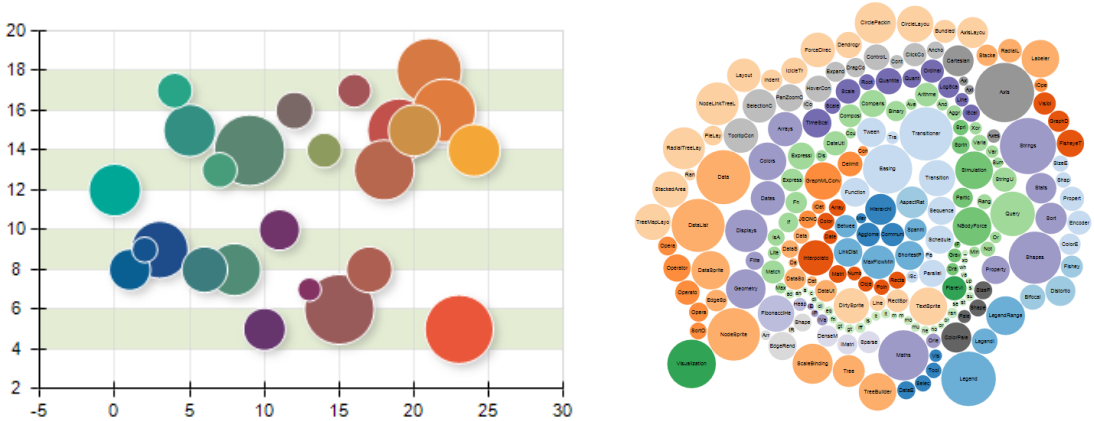
Hive diagram je unikaten graf za prikazovanje mrež različnih vrst, v katerih so vozlišča postavljena na osi glede na njihove značilnosti [8]. Število teh osi je od tri ali več, in lahko delijo vozlišča v segmente, ki jih uporabnik poljubno določi. Segmenti so po navadi urejeni

glede na gostoto ali stopnjo povezljivosti posameznih vozlišč. Zaradi teh zakonitosti so različne vizualizacije s hive diagrami ne glede na velikost podatkov zelo uniformirane. Namen diagramov je urejeno predstaviti večja omrežja, kar nam pomaga pri raziskovanju in analizi njihove strukture.



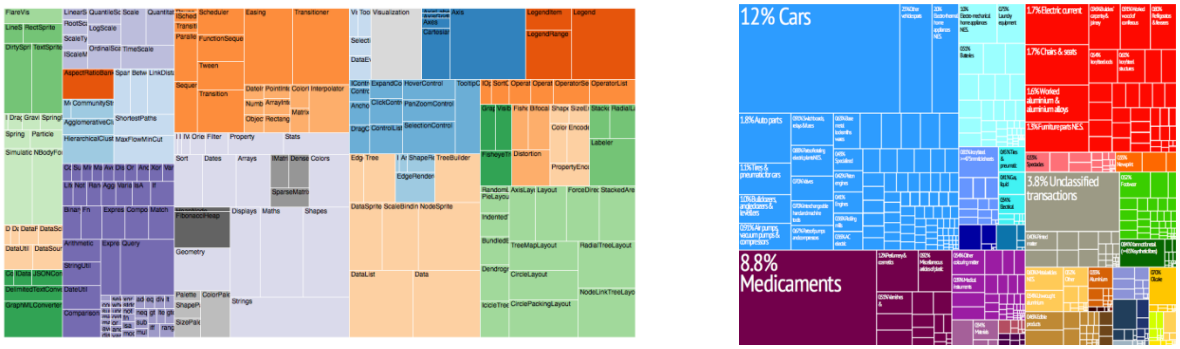
Slika 6: Dva različna hive diagrama

Graf z mehurčki ali bubble chart je naprednejša oblika razsevnega diagrama, ki namesto točk uporablja različno velike diske, kar predstavlja dodatno podatkovno dimenzijo [9]. Tako imamo vsega skupaj tri spremenljivke, ki jih lahko z določenim elementom upodobimo. To sta poziciji na koordinatah x in y ter magnituda. Pomembno je, da smo pri računanju velikosti diskov še posebej pozorni, saj človeško oko kroge primerja glede na velikost površine in ne na podlagi velikost radija. Zaradi tega bi uporaba tretje spremenljivke za definicijo radija povzročila nelinearen prikaz magnitud. Grafi z mehurčki so primerni za vizualizacijo podatkov v gospodarstvu, medicini in drugih znanstvenih branžah.



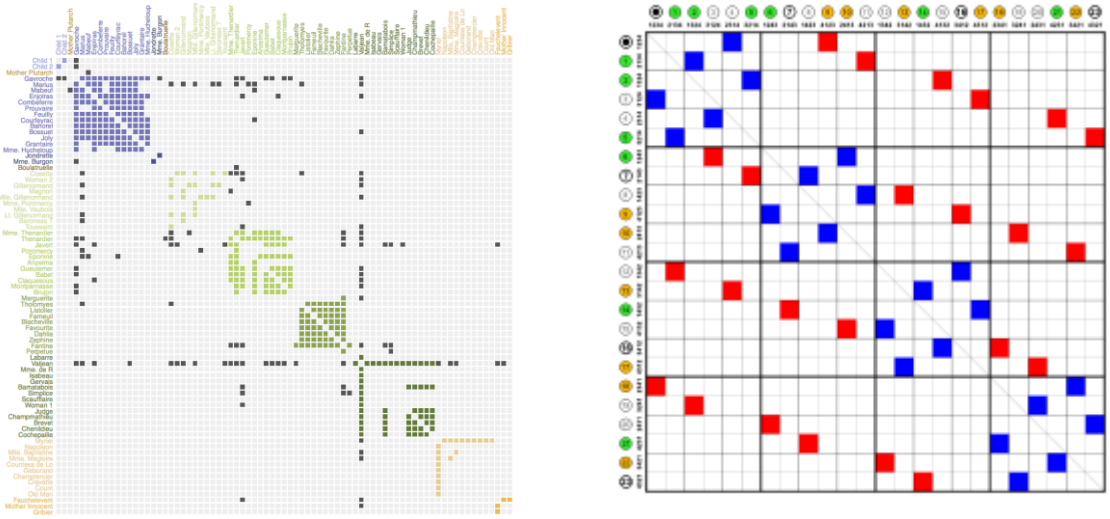
Slika 7: Dva primera diagramov z mehurčki

Treemap je način, s katerim prikazujemo hierarhične podatke z gnezdenimi kvadrati. Začnemo tako, da najprej zgradimo drevesno podatkovno strukturo, nato pa za vsako vejo narišemo kvadrat. Tega napolnimo z manjšimi kvadrati, ki predstavljajo podveje. Postopek rekurzivno ponavljamo, dokler ne obdelamo vseh vej in pridemo do najnižjega nivoja. Z barvami lahko prikažemo še dodatno podatkovno dimenzijo, kar nam olajša prepoznavanje vzorcev. Vizualizacija je zelo prostorsko učinkovita, zato je namenjena prikazu večjih podatkovnih naborov. Razvita je bila z namenom prikaza zasedenosti trdih diskov glede na velikost dokumentov [9]. Njihova najpogostejša uporaba je v gospodarstvu, pri prikazovanju porazdelitve proračuna in pri pregledu porabe sredstev.



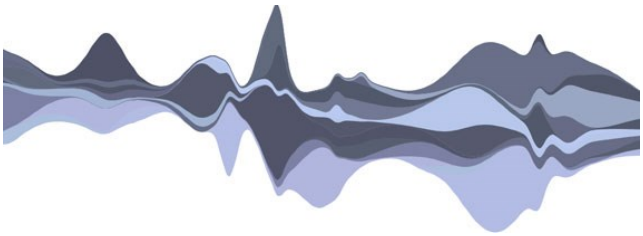
Slika 8: Diagram treemap

Matrični diagram je manj kompleksen grafičen prikaz v obliki matrike, ki zelo jasno prikazuje relacije med posameznimi elementi. Ti so razporejeni horizontalno in vertikalno, tako da med seboj tvorijo raster, znotraj katerega so s točko ali kvadratkom prikazane morebitne povezave elementov. Slabost takšnega diagrama je velika prostorska potratnost, zato je njegova izbira smiselna kadar želimo ne pregleden način prikazati relacije "many-to-many". Matrični diagrami se uporabljajo v arhitekturi za pravilno razporeditev prostorov, v računalništvu pri optimizaciji podatkovnih baz in v podjetjih za izboljšavo produktnih linij ter organizacije [10].



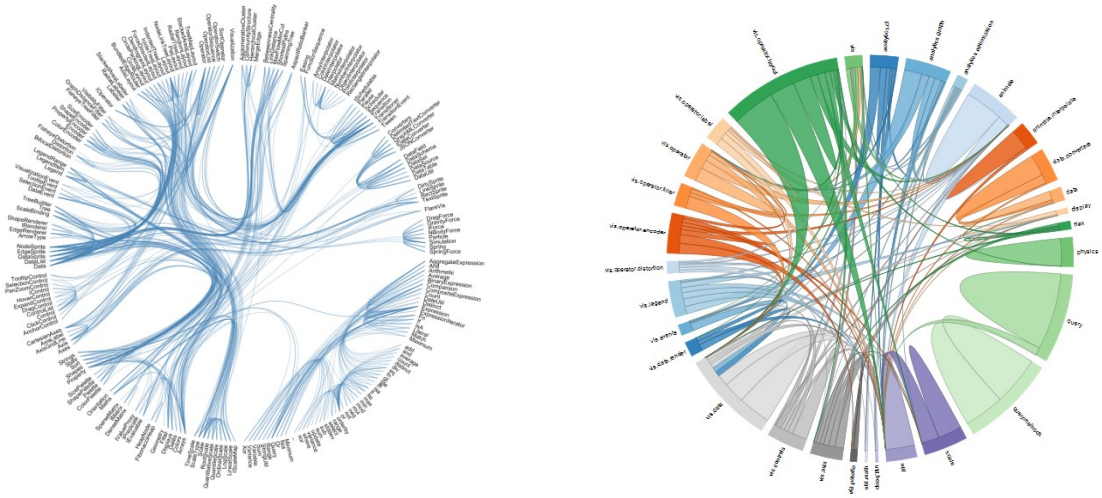
Slika 9: Matrični diagram

Streamgraph ali tokovni graf je modernejša vrsta zloženega "area" diagrama. Njegova namembnost je prikaz sprememb kumulativ več vrst spremenljivk skozi čas okoli centralne koordinate. Te lastnosti mu omogočajo izredno tekoč, plastičen in estetski izgled, zato je streamgraph primeren za predstavitev večjega nabora podatkov širši javnosti. Pojavil se je leta 2008 v vizualizaciji filmskih zaslužkov [11][12].



Slika 10: Streamgraph

Hierarchical edge bundling − graf prikazuje relacije med urejenimi elementi v krogu. S posebnim algoritmom združuje povezave v snope in tako ustvari bolj pregledno predstavitev. V vizualizacijo lahko dodamo še možnost interakcije izbire posameznega elementa. Ko ga uporabnik izbere, se prikažejo samo njegove povezave. Prednosti diagrama so prostorska učinkovitost, veliko možnosti za interakcijo in pregleden prikaz elementov.



Slika 11: Diagram hierarchical edge bundling

2.2 Izbira orodja za izdelavo podatkovnih vizualizacij

Ko gradimo spletne vizualizacije, pogosto uporabljamo več orodji hkrati. To še posebej velja za svetovni splet, kjer interaktivne vizualizacije gradimo s tehnologijami HTML (za vsebino strani), CSS (za estetiko in obliko), JavaScript (za interakcijo) ter SVG (za risanje vektorskih grafik). Eden izmed velikih uspehov spleta kot platforme je zelo dobro povezovanje zgoraj omenjenih tehnologij preko souporabe dokumentno objektnega modela (DOM). DOM podaja hierarhično strukturo vsebine spletne strani z odstavki, tabelami in drugimi elementi dokumenta. To omogoča njihovo sklicevaje in manipulacijo. Poleg programskih vmesnikov moderni brskalniki vključujejo tudi močna grafična orodja za razvijalce, ki lahko prikažejo drevesa elementov, dedovane vrednosti stilov in omogočajo razhroščevanje interaktivnih programskih skript. Na žalost pa se ta interoperabilnost izgubi v vizualizacijskih ogrodjih zaradi enkapsulacije DOM-a. Pogosto programski vmesniki za vizualizacijo z abstrakcijo funkcij za izrisovanje, interakcijo in dostopanje do elementov dokumenta poenostavijo programiranje in implementacijo, s tem pa tudi zmanjšajo možnost za manipulacijo z obstoječim modelom. Na ta način dosežejo hitrejši razvoj vizualizacij in krajši čas učenja ogrodja na račun zanemarjanja standardov in dodatnega znanja programerja, ki bi ga lahko pri bolj dodelanih ogrodjih koristno uporabil. Če povzamemo: visoka stopnja abstrakcije zmanjšuje možnosti za opisovanje, kar omejuje svobodo pri izdelavi vizualizacij in zvišuje izvajalni čas ter obremenitve sistema. Izvajanje določenih funkcij je v takih orodjih zaradi enkapsulacije oteženo, prav tako je oteženo razhroščevanje. Taka ogrodja so primerna za izdelavo vizualizacij, ki niso preveč kompleksne, ter za bolj neizkušene uporabnike [13]. Za bolj napredne in dodelane vizualizacije pa potrebujemo ogrodje, ki programerja z abstrakcijo ne omejuje pri dostopanju do stilov in do modela, ampak mu z dodatnim naborom funkcij le-

to lajša. Skozi zadnjih nekaj let se je izkazalo, da je tako ogrodje knjižnica D3.js, ki si je v tem času ustvarila široko skupnost uporabnikov.

V nadaljevanju bomo pregledali trenutno najbolj priljubljene tehnologije, začenši z D3.js. Nadaljevali bomo s formati za opis podatkov in grafike, na koncu tega poglavja pa bomo opisali še jezik CSS.

2.2.1 D3.js

Knjižnica D3.js je zbirka funkcij, napisanih v programskem jeziku JavaScript, namenjena za izdelavo spletnih podatkovnih vizualizacij. Kratica D3 pomeni Data Driven Documents oz. podatkovno gnani dokumenti. To so dokumenti, ki jih lahko prikaže spletni brskalnik, kot na primer HTML in SVG. D3.js je torej pogon, ki poveže podatke z dokumenti. Knjižnica sledi spletnim standardom in polno izkorišča zmogljivosti brskalnika.

Značilnosti programskega vmesnika so še hitrost, podpora večjim podatkovnim naborom ter možnost animacije in interakcije. Zasnovana je tako, da se lahko vsa programska koda ponovno uporablja s pomočjo komponent in vtičnikov.

Knjižnica D3.js je odprtokodna, izdana je pod licenco BSD in je prosto dostopna na spletu, preko repozitorija GitHub.

2.2.1.1 Zgodovina in nastanek D3.js

Prvotni spletni brskalniki so prikazovali le statične strani. Netscape je leta 1996 izdal prvi brskalnik z možnostjo uporabe JavaScripta kot skriptnega jezika. Ta novost je povzročila nastanek dinamičnih, interaktivnih dokumentov. Veliko spletnih projektov in aplikacijskih vmesnikov je ta skriptni jezik uporabilo za vizualno prestavitev podatkov. Nekatere implementacije so bile boljše, druge slabše.

Leta 2005 izide prva verzija orodja Prefuse, ki je namenjeno vizualizaciji podatkov na spletu. Napisan je v programskem jeziku Java. Programi, izdelani s tem orodjem, se izvajajo kot samostojne Swing aplikacije ali kot Java Appleti. Projekt Prefuse ja odprtokoden, izdan pod licenco BSD.

Dve leti za tem izide programski vmesnik Flare, napisan v tehnologiji ActionScript. Flare vizualizacije tečejo v okolju Adobe Flash Player, ki je virtualni stroj, namenjen izrisovanju vektorjev in animaciji. Tako kot Prefuse tudi Flare deluje znotraj spletnih brskalnikov kot vtičnik.

Z napredkom tehnologije so spletni brskalniki postali vedno zmogljivejši. Izrisovanje grafik je postalo možno tudi direktno znotraj brskalnikov, brez uporabe vtičnikov.

Leta 2009 na univerzi Standford izdelajo Protovis, orodje za vizualizacijo, ki temelji na skriptnem jeziku JavaScript. Protovis se izvaja direktno v brskalniku z avtohtono tehnologijo renderirnega stroja.

Protovis je poenostavil izdelavo vizualizacij, še posebej uporabnikom, ki niso vešči programiranja. To je dosegel z abstraktnim predstavitvenim nivojem. Oblikovalec je klical ta nivo s posebno sintakso, a je zaradi tega nekonvencionalnega pristopa postalo razhroščevanje zelo oteženo.

Leta 2011 Mike Bostok, Vadim Ogievetsky in Jeff Heer objavijo knjižnico D3.js. Knjižnica deluje direktno na spletnem dokumentu. To pomeni lažje razhroščevanje, lažje testiranje in več grafičnih možnosti uporabe. Edina slabost v primerjavi s Flarom je težje učenje uporabe knjižnice.

2.2.1.2 Delovanje D3.js

Delovanje knjižnice lahko razdelimo na štiri poglavitne dele delovanja:

- Nalaganje podatkov v brskalnik.
- Povezovanje podatkov z elementi znotraj dokumenta in kreiranje novih elementov, če je to potrebno.
- Transformiranje elementov z interpretiranjem povezave in nastavljanje njihovih vizualnih lastnosti.
- Prenos elementov med stanji glede na uporabniški vnos.

Transformiranja elementov je najbolj pomemben del, saj se pri tem izvaja mapiranje. D3.js zagotavlja ogrodje za izvajanje transformacij, programer pa določi pravila za mapiranje (preslikavo). Tako na primer določi velikost prikazanih stolpcev v grafu, barve posameznih držav na zemljevidu sveta, obliko vozlišč v mreži itd. [14].

2.2.1.3 Zgradba D3.js

Kot je že bilo omenjeno, ponuja knjižnica D3.js veliko funkcij za izdelavo grafov in interakcijo. Če jo želimo uporabiti za implementacijo vizualizacij, je dobro poznati njeno notranjo organizacijo. Zato bomo v tem poglavju prikazali, kako je sestavljena.

Knjižnica D3.js je razdeljena na osem glavnih delov. Osrednji del (**Core**) vsebuje osnovne funkcije za delo z elementi dokumenta. V njem se nahajajo tudi gradniki za časovne manipulacije, matrike, funkcije, ki skrbijo za nalaganje podatkov ter pretvarjanje števil, in objekti za lokalizacijo. Sledijo še nabor funkcionalnosti za podporo formatu XML in algoritmi za upravljanje z barvami ter barvnimi shemami. Paket vsebuje tudi pomemben podatkovni tip Math za razne matematične izračune.

Za delo z realnimi števili, diskretnimi vrednostmi ali za uporabo časovnic pa uporabimo paket **Scales**. Zelo pomemben del knjižnice D3.js je tudi paket **SVG,** ki nam pomaga pri risanju geometrijskih oblik, kontrol in osi (koordinat). Pri oblikovanju diagramov imamo na voljo obsežen paket **Layouts**. Tu se nahajajo objekti za postavitev in razporeditev grafov. Ti nam pomagajo pri definiciji tipa grafa z algoritmi za povezovanje vozlišč v drevesih, izdelavo dendrogramov, prikazovanje relacij med entitetami v grupah itd. Potem so tu še paket **Geo,** ki vsebuje funkcije za geografsko vizualizacijo, paket **Behaviors,** ki skrbi za interakcijo z vizualizacijo, paket **Geometry** za izvajanje poligonskih računskih operacij in paket **Time,** v katerem so zajeti vsi objekti in funkcije za delo s časom [15].

Za zajemanje podatkov knjižnica D3.js uporablja format JSON, ki ga bomo spodaj opisali.

2.2.2 JSON

JSON ali JavaScript Object Notation je tekstovni format za prenos urejenih podatkov med programskimi jeziki in aplikacijami. Izhaja iz objektnih literalov, ki so definirani v standardu ECMAScript Programing Language. JSON uporablja sintakso zavitih oklepajev, oglatih oklepajev, dvopičij in vejic za definicijo podatkovnih lastnosti in vrednosti. Definira lahko štiri primitivne tipe (niz, število, boolean, null) ter dva strukturirana tipa (objekt, array). Format JSON prepozna samo štiri vrst presledkov in ne dovoljuje dodajanja komentarjev. Prednost takšnega formatiranja podatkov v primerjavi z uveljavljenim standardom XML je manjša velikost dokumenta in hitrejše nalaganje [16][17].

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Primer 1: Izgled formata JSON

2.2.3 GeoJSON

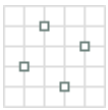
Je razširitev formata JSON, namenjena je kodiranju različnih geografskih struktur. Objekti v GeoJSON-u lahko predstavljajo geometrijo, funkcijo ali kolekcijo funkcij. Format podpira naslednje podatkovne tipe: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, in GeometryCollection. GeoJSON se od ostalih GIS standardov razlikuje v tem, da ga ne pišejo in vzdržujejo priznane organizacije za standardizacijo, ampak skupina spletnih razvijalcev.

Omeniti je treba še TopoJSON, podaljšek GeoJSON-a, ki se uporablja za opisovanje topologij oz. razmerji med točkami, linijami in poligoni, s katerimi predstavljamo značilnosti geografskega območja. Pomembna lastnost tega formata je manjša velikost in boljša portabilnost znotraj knjižnice D3.js [18].

Point

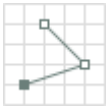

```
{
    "type": "Point",
    "coordinates": [30, 10]
}
```

MultiPoint



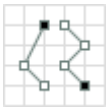
```
{
    "type": "MultiPoint",
    "coordinates": [
       [10, 40], [40, 30], [20, 20], [30, 10]
    ]
}
```

LineString



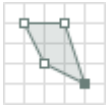
```
{
    "type": "LineString",
    "coordinates": [
       [30, 10], [10, 30], [40, 40]
    ]
}
```

MultiLineString



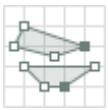
```
{
    "type": "MultiLineString",
    "coordinates": [
        [[10, 10], [20, 20], [10, 40]],
        [[40, 40], [30, 30], [40, 20], [30, 10]]
    ]
}
```

Polygon



```
{
    "type": "Polygon",
    "coordinates": [
    [[30, 10], [40, 40], [20, 40], [10, 20],
[30, 10]]
    ]
}
```

MultiPolygon



```
{
    "type": "MultiPolygon",
    "coordinates": [
        [
        [[30, 20], [45, 40], [10, 40], [30, 20]]
        ],
        [
        [[15, 5], [40, 10], [10, 20], [5, 10],
[15, 5]]
        ]
    ]
}
```

Primer 2: Zgoraj so prikazani geometrijski tipi v formatu GeoJson

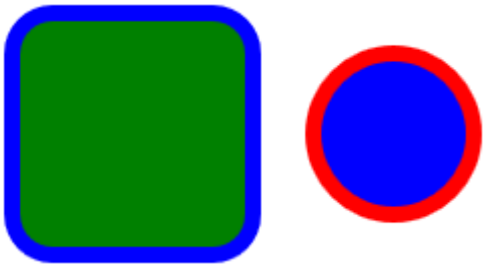
Pri risanju vektorskih grafov, D3.js uporablja standard SVG. V naslednjem poglavju si ga bomo na kratko ogledali in podali primer kode.

2.2.4 SVG

Scalable Vector Graphics je standard za opisovanje dvodimenzionalnih grafik v tehnologiji XML. Omogoča izrisovanje črt, krivulj, slik in teksta. Vse grafične objekte lahko združujemo, stiliziramo, transformiramo in sestavljajo v predizrisane oblike. Nabor funkcij vključuje tudi gnezdene transformacije, maskiranje, filtrske efekte in raznovrstne predloge objektov.

SVG grafike so lahko interaktivne in dinamične. Animiramo jih s skriptami ali z uporabo deklaracij. Napredna uporaba SVG-ja je možna z nadomestnim skriptnim jezikom, ki dostopa do dokumentno objektnega modela (DOM). Na ta način je možno naslavljati vse elemente, atribute in lastnosti. Vmesnik vsebuje tudi bogat seznam kontrol dogodkov, kot na primer »onmouseover« in »onclick«, ki se lahko dodelijo poljubnemu grafičnemu SVG objektu. Zaradi kompatibilnosti s spletnimi standardi je možno istočasno izvajanje skript nad elementi XHTML in SVG [19].

SVG je odprt standard, razvit v World Wide Web Konzorciju (W3C) leta 1999. Podpirajo ga vsi pomembnejši spletni brskalniki: Mozilla Firefox, Internet Explorer, Google Chrome, Opera, in Safari.



```
<svg x="10">
      <rect x="50" y="10" rx="20" ry="20" width="120" height="120"
      style="fill:green;stroke:blue;stroke-width:8;opacity:1" />
</svg>

<svg x="200">
      <circle cx="50" cy="70" r="40" stroke="red" stroke-width="8"
      fill="blue" />
</svg>
```

Primer 3: Programska koda, ki izriše zaobljen kvadrat in krog

2.2.5 CSS

Cascading Style Sheets je jezik, ki je bil razvit v CERN-u leta 1996. Namenjen je opisovanju stilov, ki se uporabljajo pri oblikovanju dokumentov HTML, XHTML, XML in pri drugih sorodnih formatih. Dandanes velika večina spletnih strani uporablja ravno CSS.

Stil je v CSS pravilo, ki definira barve, obliko, tekst, odmike in ostale elemente dokumenta. Pomembna prednost jezika je ločevanje vsebine od oblike, kar olajšuje vzdrževanje spletnih strani in omogoča uporabo istega oblikovanja na več podstraneh. Prav tako omogoča, da se isti dokument predstavi z različnimi stili. S tem se ga prilagodi drugačnim prikazovalnikom, bodisi zaslonom (različnih resolucij), tiskalnikom, Braillovim napravam itd. Jezik CSS vzdržuje in razvija konzorcij W3C.

Sintaksa jezika CSS je dokaj enostavna. Sestavljena je iz selektorjev, lastnosti in vrednosti. Selektor je del stila, ki pove, kateri znački (tag) v dokumentu oziroma njegovi identifikaciji, razredu ali atributu želimo aplicirati lastnosti. Za tem sledi zaviti oklepaj, znotraj katerega se stil dejansko definira. To storimo z deklaracijo lastnosti (rezervirano besedo) in z izbiro vrednosti. Deklaracij lastnosti in vrednosti je lahko znotraj selektorja več, ločeni pa morajo biti s podpičji. Ena izmed pomembnih značilnosti CSS-a je tudi dedovanje, s katerim se stili priredijo tudi gnezdenim elementom. CSS koda se lahko nahaja znotraj značke elementa, lahko je v svojem dokumentu ali pa se nahaja v glavi dokumenta.

```
#front {
  display: inline;
  font-size: 100%;
}
div.figure {
  float: right;
  width: 30%;
  border: thin silver solid;
  margin: 0.5em;
  padding: 0.5em;
}
div.figure p a{
```

```
  text-align: center;
  font-style: italic;
  font-size: smaller;
  text-indent: 0;
}
img.scaled {
  width: 100%;
}
```

Primer 4: Koda v jeziku CSS dodaja stile definiranim elementom

2.3 Druge rešitve za vizualizacije v spletu

Vsekakor D3.js ni ne prva ne edina rešitev za vizualizacijo. Že pred njenim nastankom je obstajalo veliko tehnologij, ki so se posvečale temu problemu. Najbolj priljubljen je bil Adobe Flash, ki je s svojim integriranim razvojnim okoljem, programskim jezikom ActionScript in močnim virtualnim strojem uspešno reševal probleme pri programiranju bolj kompleksnih vizualizacij. Spodaj je opisana omenjena tehnologija, vključno z razširitvijo flare, ki vsebuje dodatna orodja za risanje grafov in izvajanje interakcij.

2.3.1 Flash

Je multimedijska in programska platforma v lasti podjetja Adobe Systems. Namenjena je oblikovanju vektorskih grafik, animaciji, izdelavi računalniških igric in programiranju bogatih spletnih aplikacij (RIA), ki se izvajajo znotraj Flash predvajalnika. Flash se pogosto uporablja na svetovnem spletu za predvajanje videa, glasbe in za prikazovanje dinamičnih oglasov.

Flash z manipulacijo vektorjev in rastrske grafike zagotavlja animacijo teksta, risb in statičnih slik. Omogoča oboje smerni prenos zvoka in videa. Aplikacije lahko izdelamo direktno znotraj razvojnega okolja Adobe Flash Professional ali pa jih sprogramiramo z uporabo objektnega skriptnega jezika ActionScript.

Predvajalnik Adobe Flash Player je na voljo brezplačno za večino operacijskih sistemov in spletnih brskalnikov v obliki vtičnika. Za mobilne telefone, tablice in elektronske naprave pa je pri določenih konfiguracijah na voljo kot Flash Lite.

Za prenos podatkov lahko uporablja poleg XML in JSON tudi format AMF (Action Message Format), s katerim serializira objektne grafe.

Dokumente lahko shranimo v obliki formata SWF (ShockWave Flash). Poglejmo še njegovo zasnovo. Glavni primitivni tip v formatu je pot (path), ki predstavlja niz segmentov linij in bezirejevih krivulj. Dodatni primitivni tipi pa so kvadrati, elipse in tekst. Grafični elementi v

SWF so torej precej podobni formatoma SVG in MPEG-4 BIFS. V sodobnejših verzijah podpira tudi zvok in video. V verziji predvajalnika Flash Player11 je Adobe predstavil nizkonivojski 3D grafični programski vmesnik z imenom Stage3D, ki je soroden ogrodjem OpenGL, Direct3D in WebGL. Še zanimivost: senčenje je v tem programskem vmesniku opisano z jezikom AGAL (Adobe Graphics Assembly Language) [20] [21] [22] [23].

2.3.2 ActionScript

ActionScript je objektno orientiran programski jezik, ki so ga razvili pri podjetju Macromedia (sedaj v lasti Adobe Systems). Je dialekt ECMAScript, kar pomeni, da uporablja podobno sintakso in semantiko kot JavaScript. Namenjen je razvoju spletnih strani in programske opreme, ki se izvaja znotraj okolja Flash v obliki vgrajenih SWF dokumentov.

Različica ActionScript3 se uporablja tudi pri platformi Adobe Integrated Runtime system (AIR) za razvoj namiznih in mobilnih aplikacij.

Sam jezik je odprtokoden, njegova specifikacija pa je na voljo brezplačno. Prav tako sta odprtokodna prevajalnik (kot del Apache Flex) in virtualni stroj (Mozilla Tamarin).

Prvotno je bil ActionScript razvit za kontrolo dvodimenzionalnih vektorskih grafik, narejenih v tehnologiji Flash. Začetne verzije so bile zelo omejene, saj se jih je lahko uporabljalo zgolj za kontrolo animacij in programiranje interaktivnih funkcij. V kasnejših verzijah pa so mu dodali nove funkcionalnosti za implementacijo spletnih igric in bogatih spletnih aplikacij (RIA). Med drugim se danes ActionScript uporablja tudi za delo s podatkovnimi bazami in pri osnovnem programiranju robotike z dodatnim ogrodjem Make Controller Kit.

Knjižnice v Flashu delujejo tudi v povezavi z jezikom XML in na ta način dostavljajo bogato vsebino spletnim brskalnikom. Ta konfiguracija se imenuje Asynchronous Flash in XML, kar je podobno kot AJAX. Za izdelavo bogatih spletnih aplikacij s skriptnim jezikom ActionScript je na voljo aplikacijski programski vmesnik (SDK) Flex, ki ga razvijajo pod organizacijo Apache Software Foundation.

ActionScript sestoji iz fundamentalnih oziroma osnovnih podatkovnih tipov, ki se uporabljajo za izdelavo drugih, bolj kompleksnih in so zelo podobni podatkovnim tipom v jeziku Java.

Za razliko od nekaterih objektnih orientiranih programskih jezikov ActionSript ne razlikuje med primitivnimi in referenčnimi tipi. Znotraj njega so vse spremenljivke referenčne, čeprav so vsi objekti, ki pripadajo primitivnim podatkovnim tipom (Boolean, Number, int in String), fiksni.

2.3.3 Flare

Flare je knjižnica, namenjena izdelavi vizualizacij, ki se izvajajo v predvajalniku Adobe Flash Player. Orodje podpira izdelavo osnovnih grafikonov, grafik, interaktivnih vizualizacij in omogoča upravljanje s podatki, vizualno kodiranje, animacijo, interakcijo ter modularen razvoj vizualizacijskih funkcij. Programska koda vmesnika je odprta in izdana pod licenco BSD.

Knjižnico sestavlja deset paketov, ki jih bomo zaradi boljšega razumevanja na kratko našteli in opisali:

- Analytics: namenjen je za statistiko in analizo podatkov.
- Animate: v njem se nahajajo orodja za izdelavo in izvajanje animacij.
- Data: vsebuje metode za branje in pisanje podatkovnih naborov.
- Display: paket uporabljamo za delo z objekti DisplayObject, ki deduje objekt flash.display.
- Flex: je ovoj (wrapper), ki skrbi za izvajanje Flare aplikacij v okolju Flex.
- Physics: je koristna zbirka funkcij za fizikalne izračune.
- Guery: v paketu je procesor za pisanje poizvedb po ActionScript objektih.
- Scale: sestavljajo ga funkcije za upravljanje s podatkovnimi nivoji (scales).
- Util: nabor raznih koristnih algoritmov.
- Vis: vsebuje različne komponente(tipa flare) za vizualizacijo.

V nadaljevanju bomo predstavili osnove delovanja knjižnice flare.

Razred Transition je temeljni razred za vse animacije v knjižnici. Skrbi za njihovo izvajanje, ki je lahko istočasno ali posamično, po vnaprej določenem vrstnem redu. Objekti, ki jih uporabljamo za delo z animacijami, so Tween, Sequence, in Parallel.

Serijske animacije s Transitioners

Z njimi lahko ustvarimo tudi poljubno število animiranih prehodov. Pri delu z veliko objekti naenkrat (kar je pri vizualizaciji pogosto) postane njihova uporaba težavna. Za reševanje teh

težav Flare zagotavlja razred Transitioner. Ta poenostavlja proces izdelave animacij za zbirko objektov, tako da jim enemu za drugim priredi poljubne parametre. V ozadju Transitioner samodejno generira in ponovno uporabi potrebne "tweene" in z njimi zgradi celotno animacijo. Če objektov ne želimo animirati, jim lahko namesto tega z objektom Transitioner nastavimo le druge specifične vrednosti. Z drugimi besedami: Transitioner predstavlja vmesni nivo za osveževanje lastnosti objektov. Te osvežitve se nato zbere skupaj in animira ali aplicira posamezno.

Osnovna predstavitev podatkov v Flaru uporablja vgrajene podatkovne tipe Object in Array. Podajmo primer: tabela s podatki se implementira z »arrayem« objektov, ki vsebujejo imena in vrednosti v posameznem podatkovnem polju. Čeprav so možni tudi drugi, bolj učinkoviti pristopi, je ta način najbolj fleksibilen, obenem pa uporablja tudi že obstoječo konvencijo za razvoj v okolju Flash.

Podatke lahko v ogrodju Flare nalagamo na več načinov. Najbolj enostaven način je, da jih kar vgradimo v samo aplikacijo. S tem se podatki prenesejo hkrati ob prenosu aplikacije, kar je optimalno za statične podatkovne nabore. Drug način pa je, da jih naložimo z uporabo ActionScript objektne notacije, ki podatke definira kot spremenljivko.

```
package {
    import flash.display.Sprite;

    [SWF(width="800", height="600", backgroundColor="#ffffff",
      frameRate="30")]
    public class Test extends Sprite
    {
        public function Test()
        {
            var sprite:Sprite = new Sprite();

            sprite.graphics.beginFill(0xcccccc, 0.5);
            sprite.graphics.lineStyle(1, 0x000000);
            sprite.graphics.drawCircle(0, 0, 10);

            this.addChild(sprite);

            sprite.x = 50;
            sprite.y = 50;
            trace("our sprite is at: "+sprite.x+", "+sprite.y);
        }
    }
}
```

Primer 5: Programska koda, ki izriše krog

V večini primerov pa jih nalagamo dinamično, bodisi direktno s spletne strani (z uporabo JavaScripta) ali preko strežnika. Takih pristopov je lahko več. Najbolj pogosti so z uporabo

standarda E4X (ECMAScript for XML), ki omogoča preslikavo XML podatkovnih tipov v jezik ECMAScript [24]. Knjižnica Flare lahko podatke uvaža še preko dokumentnih formatov DSV, JSON in GraphML (XML format za prezentacijo mrež v obliki vozlišč in povezav) s pomočjo objekta DataSource [25].

2.4 Ostale manjše knjižnice za delo z grafi in zemljevidi

Za namene vizualizacij na svetovnem spletu obstaja veliko knjižnic. Nekatere so zelo zmogljive in razširjene, druge pa so še vedno v razvojni fazi. Razlikujejo se v namembnosti in programski zasnovi, skupno pa imajo uporabo jezika JavaScript in odprto kodo. V naslednjem poglavju se nahaja pregled in kratek opisi najbolj priljubljenih alternativnih knjižnic.

Google Chart Tools
Knjižnica se je razvila iz Image Charts API in temelji na HTML5/SVG tehnologiji. Z ogrodjem lahko izdelamo klasične grafikone, ki so lahko interaktivni in se jih da tudi približevati. Implementira se jih s programskim jezikom JavaScript, podatke pa nalagamo s pomočjo vgrajenega razreda DataTable.

gRaphael
Orodje odlikuje podpora starejšim spletnim brskalnikom. Je dokaj osnovno in enostavno za uporabo. Trenutna verzija vsebuje razrede: Element, Paper(), dotchart(), g(), linechart().

Highcharts JS
Je programski vmesnik v JavaScriptu, ki vsebuje že izdelane vizualizacije, grafikone, grafe in zemljevide. Komercialna različica je plačljiva, za osebno rabo pa je na voljo brezplačno. Izvorna koda je na voljo preko repozitorija GitHub.

JavaScript InfoVis Toolkit
Tako kot Highcharts JS, tudi InfoVis ponuja veliko že pripravljenih vizualizacij in primerov. Opis nabora funkcij je zelo skop, tako da je uporabnik v veliki meri prepuščen raziskovanju.

jqPlot
Je vtičnik za jQuery, ki ponuja osnovne grafikone. Je dokaj enostaven za uporabo in ima zelo dobro dokumentacijo podprtih razredov. Na spletni strani so tudi uporabniška navodila in tutoriali. Vtičnik je odprtokoden in na voljo brezplačno.

jQuery Sparklines
Je vtičnik za jQuery. Namenjen je izdelavi t. i. sparklines – zelo majhnih črtnih grafikonov, ki se jih uporablja v vrsticah, med tekstom. Kompatibilen je tudi s starejšimi brskalniki. Na

spletni strani projekta se nahaja forma, s katero se lahko izdela graf in doda podatke, prav tako se tam nahaja tudi obširna dokumentacija. Koda je izdana pod licenco BSD.

Peity

Je podobna rešitev kot jQuery Sparklines, prav tako implementirana kot vtičnik za jQuery. Odlikuje jo izredno kompaktna programska koda. Podpira samo novejše brskalnike, navodila za uporabnike pa so zelo skopa.

Timeline.js

Je odprtokodno orodje za izdelavo interaktivnih časovnih diagramov, napisano v JavaScriptu. Na voljo je v 40 svetovnih jezikih. Vgrajeno ima podporo za Twitter, Flickr, Google Maps, Youtube, Vimeo, Dailymotion, Wikipedio in ostale popularne spletne strani in servise. Časovni diagram se lahko izdela zelo enostavno, s spletnim čarovnikom in s podatki iz Google spreadsheet.

arbor.js

Je knjižnica, napisana z uporabo jQueryja, namenjena predvsem dinamičnemu prikazovanju mrež. Izvaja se lahko le v modernejših brskalnikih. Zagotavlja force-directed layout algoritem, abstrakcijo za organizacijo grafov in funkcije za osveževanje zaslona.

Sigma js

Tako kot arbor.js, je tudi ta knjižnica prilagojena za izdelavo dinamičnih mrež. Napisana je v JavaScriptu, podatke pa zajema iz dokumentov json. Sigma ima vgrajeno podporo za Canvas in WebGL. Odlikujeta jo izredna hitrost in visoka prenosljivost.

Kartograph

Je aplikacijski programski vmesnik, implementiran v programskih jezikih JavaScript in Python. Namenjen je programiranju stiliziranih, interaktivnih zemljevidov, brez uporabe aplikacij kot Google Maps ali kakršnih koli drugih servisov za mapiranje. Izdelan je bil še posebej za oblikovalce in novinarje. Na spletni strani projekta je veliko primerov in navodil. Koda je izdana pod licenco A/L GPL in je dostopna preko GitHuba.

Leaflet

Je moderna, odprtokodna knjižnica, prilagojena mobilni in spletni rabi. Razvita je z mislimi na enostavnost, performaso, uporabnost in portabilnost. Uporablja tehnologiji HTML5 in CSS3, deluje v vseh novejših brskalnikih. Še posebej je namenjena razvijalcem on-line zemljevidov. Njena zelo pomembna vrlina je integracija z OpenStreetMap (odprtokodnim projektom za razvoj zemljevida sveta).

Modest Maps
Tako kot že samo ime pove, gre za izredno okleščeno, portabilno in lahko knjižnico za delo s svetovnimi zemljevidi.

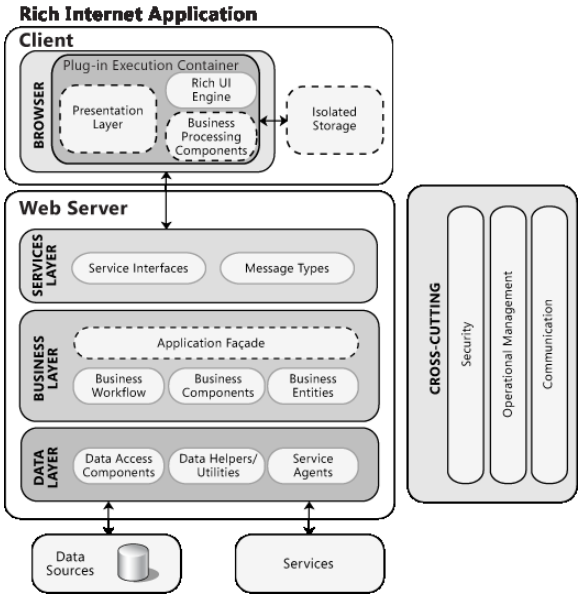
Omeniti velja še knjižnice za delo s 3D grafiko, in sicer PhiloGL in three.js, ter knjižnice, narejene z ogrodjem D3: Crossfilter, Cubism, Dashku, dc.js, NVD3, Rickshaw, Tributary itd.

Če bi se programer želel odločiti za eno izmed teh knjižnic, bi bila trenutno najbolj smotrna odločitev izbira knjižnice Google Chart Tools, ki ima obširno in kvalitetno dokumentacijo ter veliko število grafov, ki so sicer bolj osnovne oblike. Prav tako dobre izbire za hiter prikaz podatkov sta tudi knjižnici jQuery Sparklines ali Peity, ki sta namenjeni predvsem prikazovanju majhnih grafov med tekstom. Za delo z zemljevidi pa prideta v poštev najbolj dodelana vmesnika: Kartograph in Leaflet.

2.5 Bogate spletne aplikacije − Rich Internet Applications (RIA)

So aplikacije, ki se izvajajo v brskalniku, pogosto preko vtičnikov in imajo značilnosti okenskih aplikacij. Njihovi vmesniki omogočajo uporabo naprednih in bolj dodelanih grafičnih uporabniških komponent. So bolj odzivne in interaktivne kot navadne spletne aplikacije. To so razlogi, da se jih imenuje bogate spletne aplikacije. Po navadni je za njihovo poganjanje potrebna namestitev dodatne programske opreme, ki je v večini primerov kar virtualni stroj, ki se preko vtičnikov izvaja v brskalnikih. Njihova prednost je, da generiranje strani in logika za interakcijo tečeta na strani klienta. Prevladujejo pri programih z visoko stopnjo interakcije in grafike, kot na primer spletne igrice.

Najbolj pogoste tehnologije za izvajanje bogatih spletnih aplikacij so Adobe Flash, Microsoft Silverlight, Java Applets in JavaFX.



Slika 12: Diagram arhitekture RIA aplikacij[26]

2.6 Infografike

Beseda infografika izhaja iz informacije in grafike. Infografika lahko rečemo vsaki grafični obliki, ki podaja neko informacijo. Tako gre lahko za preprost linearni graf ali za zelo kompleksno podatkovno vizualizacijo. Bolj točno pa beseda zaznamuje večje grafične prezentacije, ki združujejo grafe, diagrame, tekst, ikone, podatkovne vizualizacije, slike in simbole v neko skupno zgodbo. Zaradi tega se lahko iz njih razbere širši kontekst prikazanih podatkov. Infografike so pogosto prave likovne umetnine, pri katerih se pozornost še posebej posveča izbiranju barv, tipografiji, simetriji elementov in kompoziciji.

Pregled infografik skozi zgodovino[27]

- 6200 pr.Kr. Najstarejši zemljevid.
- 500 pr. Kr. Prvi zemljevid sveta.
- 150 Prvič se uporabi zemljepisna širina in višina − Claudius Ptolemy.
- 950 Prvi primer grafične uporabe spreminjajočih se vrednosti (pozicija Sonca, Lune in planetov skozi leto).
- 1305 Mehanski diagram znanja.
- 1350 Francoski škof Nicole Oresme predlaga uporabo stolpcev za predstavitev različnih vrednosti.
- 1626 Vizualizacija sprememb sončnih peg − Christopher Scheiner.
- 1644 Prva vizualizacija statističnih podatkov – razlike v določanju dolžine med Toledom in Rimom − Michael F. van Langren.

- 1753 Carte chronologique – kronološka karta – prvi časovni trak − Jacques Barbeu-Dubourg.
- 1758 diagrami barvnih sistemov − Johanes Tobias Mayer.
- 1765 Časovnica Življenje 2000 pomembnih ljudi − Joseph Priestley.
- 1779 Grafična analiza spreminjanja temperature zemlje − Johann Heinrich Lambert.
- 1782 Uporaba proporcionalnih kvadratov za predstavitev demografskih podatkov − Charles de Fourcroy.
- 1786 Vizualizacija ekonomskih podatkov − William Playfair.
- 1801 Prvi tortni diagram − William Playfair.
- 1817 Vizualizacija povprečnih temperatur po svetu − Alexander von Humboldt.
- 1819 Prikaz stopenj nepismenosti po regijah, na zemljevidu − Pierre Charles Dupin.
- 1829 Vizualizacija kriminala − Andr´e-Michel Guerry.
- 1844 Tableau-graphique – Prikaz komercialnega transporta − Joseph Minard.
- 1851 Vizualizacija proizvodnje premoga po regijah, na zemljevidu − Charles Joseph Minard.
- 1869 Diagram prikaza Napoleonovega napada na Moskvo – Ena izmed najbolj znanih vizualizacij v zgodovini − Charles Joseph Minard.
- 1873 Prvi prikaz s tabelo, v kateri so podatki podani z različnimi stopnjami senčenja − Toussaint Loua.
- 1896 Uporaba kvadratov na zemljevidu, ki prikazujejo dve spremenljivki v dveh dimenzijah − Jacques Bertillon.
- 1930 Zgodovinski dogodki prikazani na logaritemskem papirju − Heinz Von Foerster.
- 1933 Diagram podzemne železnice, ki daje prednost uporabnosti − Henry C. Beck.
- 1968 Grafični vzorci za statistično vizualizacijo − Roberto Bachi.
- 1971 Biplot – metoda za vizualizacijo opažanja in spremenljivk v statistiki − Ruben Gabriel.
- 1972 Plošča na sondi Pioneer 10 – sporočilo je prikazano z infografiko – NASA.
- 1981 Prikaz podatkov v mozaiku − John Hartigan.
- 1982 Časopis USA Today z vizualizacijami vremenskih podatkov začne ero infografik.
- 1986 Avtomatsko oblikovane grafične predstavitve relacijskih podatkov − Jock Mackinlay.
- 1991 Prikaz podatkovne strukture TreeMap z gnezdenimi kvadrati − Ben Shneiderman.

Poglavje 3 Implementacija primera

Vse znanje, ki smo ga pridobili pri pregledu področja vizualizacije podatkov, bomo v tem poglavju združili v praktičen primer.

Opis problema
Imamo zbirko slovenskih ljudskih pesmi, ki so shranjene v podatkovni bazi. Zbirka vsebuje različne vrste podatkov, od zvočnih posnetkov, besedila, kraja, faktorja podobnosti do vrste snemalne opreme. Podatki so med seboj povezani in urejeni. Za osnovni pregled zbirke lahko uporabimo obstoječi informacijski sistem, ki obenem skrbi tudi za vnos in validacijo informacij. Ker želimo značilnosti slovenske ljudske glasbe še bolj spoznati in dodelati funkcionalnost informacijskega sistema, jih bomo skušali vizualizirati na čim boljši način.

Na začetku si bomo odgovorili na ključna vprašanja pri snovanju vizualizacij, ki so našteta v poglavju 2.1, in z njimi definirali robne pogoje. Torej: vizualizacijo želimo predstaviti potencialno zainteresiranim ljudem preko spletne strani. S tem bomo zagotovili dobro dosegljivost in dostopnost. Vizualizacija bo namenjena prikazu na računalniških zaslonih, velikosti od 17 do 24 palcev. Zaradi kompleksnosti prikaza in interakcije ne bomo posvečali posebne pozornosti mobilnim telefonom in tablicam, a jih bomo imeli vsekakor v mislih (bootstrap, responsive design). Prikazali bomo od dve do tri dimenzije podatkov, saj − kot je že bilo rečeno − so take vizualizacije najbolj pregledne. Način interakcije s podatkovnimi vizualizacijami bo možen z uporabo računalniške miške, pozorni pa bomo tudi na interakcijo s prsti (velikost najmanjših elementov).

Osredotočili se bomo na prikaz podobnosti pesmi, zato bomo izbrali diagram z mehurčki, ki prikazuje elemente v koordinatnem sistemu. Podobnost bomo definirali z odmiki med mehurčki in njihovo magnitudo. Bolj sorodni bodo med seboj bližje kot manj sorodni. S klikom na določen element bomo prikazali njegove vrednosti in po potrebi naložili pesem v predvajalnik zvoka, ki bo v obliki komponente HTML5.

Da bi dojeli širši kontekst umestitve ljudske glasbe v slovenski prostor, bomo pesmi prikazali tudi na zemljevidu, razdeljenem v regije. Temnejše barve regij bodo predstavljale večje število skladb. Ob kliku na poljuben mehurček v prvem diagramu se bo na zemljevidu označila tudi regija, v kateri se izbrana pesem nahaja.

Na koncu bomo iz vseh prikazanih pesmi zgradili drevo podobnosti in ga prikazali kot množico diskov, ki predstavljajo liste in veje v drevesu. Tako kot zemljevid, bomo tudi drevo iz diskov povezali z grafom mehurčkov in ob kliku z animacijo prikazali izbrani disk, ki bo

predstavljal isto pesem. Tako bomo dobili boljšo predstavo, kje v hierarhiji se določena pesem nahaja glede na podobnost.

Implementacija

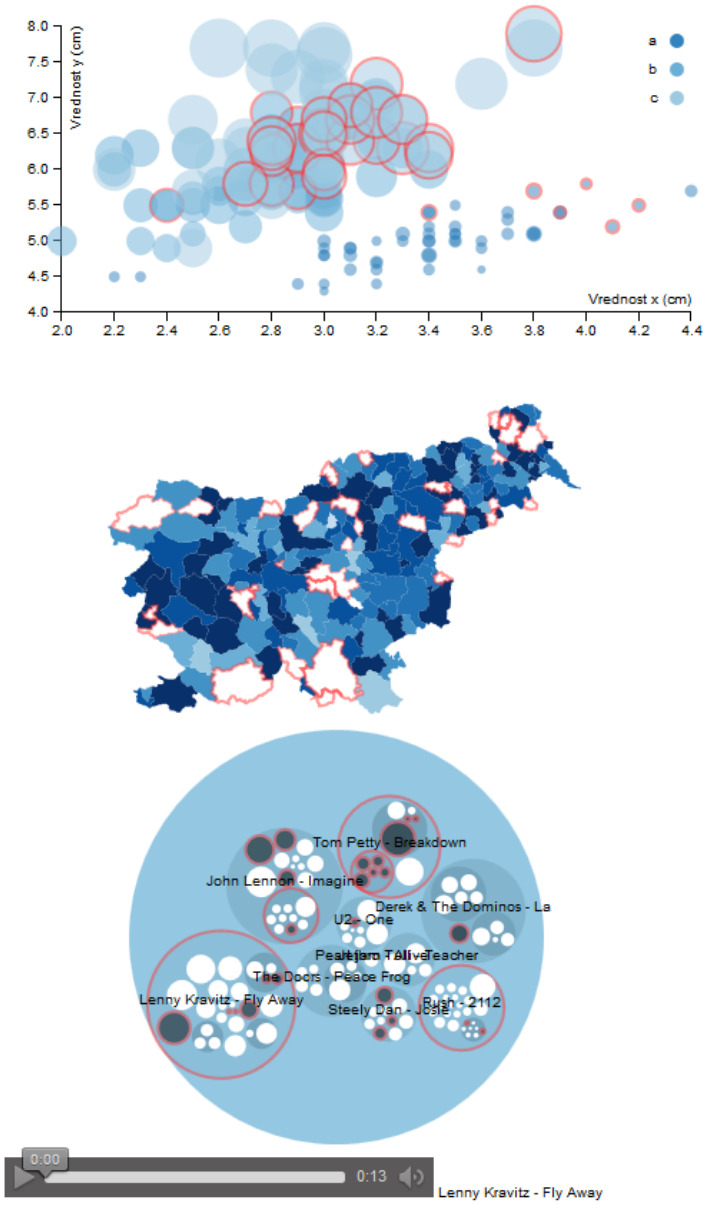
Začnimo s HTML dokumentom, ki bo nosil programsko kodo. Za izvedbo smo se odločili izbrati skriptni jezik JavaScript in mu dodali knjižnico D3.js, ki že vsebuje nekatere predloge diagramov na svoji spletni strani. Najprej definiramo njihovo velikost, ki bo dimenzije 500 x 250 pikslov. Nato za prvi graf določimo osi x in y, ki ju instanciramo kot spremenljivki. Nadaljujemo z izbiro barvne skale s funkcijo `d3.scale.category20c()`, ki vrne 20 pripravljenih barv. Podatke v formatu JSON naložimo s klicem funkcije `d3.json()`, znotraj katere se z iteracijo sprehodimo čez celoten nabor podatkov in jih z ukazom `append()` dodamo v telo dokumenta HTML. Tehnologijo SVG uporabimo za risanje in prikaz elementov v obliki diskov, tako da izberemo značko (tag) `<circle>` in njenim atributom r (radij), cx (pozicija x), cy (pozicija y) priredimo prebrane podatke. Vsakemu elementu dodamo še prisluškovalca dogodkov (event listener) za klik z miško, s katerim sprožimo prikaz meta podatkov in HTML5 komponente za predvajanje skladbe (`<audio>`).

Na vrsti je izdelava drugega diagrama, ki bo prikazoval na zemljevidu obarvane regije glede na število skladb, ki jih vsebujejo. Najprej izberemo geografsko projekcijo Albers s funkcijo `d3.geo.albers()` in ji določimo center, rotacijo, paralele in bližino (zoom). Omenjeno projekcijo smo izbrali, ker najlepše prikaže območje med vzporednikoma, med katerima se nahaja Slovenija. Za tem naložimo podatke za izris države in za jakost intenzitete barv, ki bodo predstavljale število pesmi. Uporabimo formata GeoJSON in TSV. Nato se ponovno z iteracijo sprehodimo čez vse elemente in za vsakega z značko `<path>` narišemo občino, s prejetimi koordinatami. Pri barvanju si pomagamo s kvantizacijo, ki na podlagi naložene številke jakosti intenzitete določi njeno diskretno vrednost (celo število). To storimo s funkcijo `d3.scale.quantize()`. Tudi na tem grafu dodamo prisluškovalca dogodkov (event listener) za klik z miško. Ko bomo kliknili na poljubno občino, bomo v ostalih grafih prikazali elemente (pesmi), ki se v njej nahajajo.

Zadnji graf bo narejen s t. i. pakiranjem krogov (circle packing). Prikazali bomo diske, ki so ugnezdeni glede na hierarhijo elementov, in se ne prekrivajo, ampak dotikajo. Tukaj si bomo v veliki meri pomagali s predlogo za diagram, ki se nahaja na spletni strani D3.js. Začnemo s funkcijo `d3.layout.pack()`, ki je namenjena temu tipu vizualizacije. Z njeno uporabo določimo stopnjo in premer krogov. Podatke, ki so že urejeni v drevesno strukturo, naložimo iz formata JSON z iteracijo. Ponovno jih narišemo s SVG značko `<circle>`, jim določimo velikost, ki smo jo dobili iz naloženih podatkov, in jih pobarvamo s pomočjo funkcije `d3.scale.linear()`, v kateri določimo barvni obseg. Na koncu dodamo animacijo prehodov

pri klikanju posameznih krogov. Tukaj uporabimo funkcijo `d3.interpolateZoom()` za približevanje in funkcijo `d3.transition().duration()` za definicijo hitrosti animacije. Pravilno velikost približevanja dosežemo z uporabo posebnega izračuna.

Na koncu je treba še povedati, da si za izbiranje elementov v različnih grafih pomagamo tako, da vsakemu od njih priredimo unikaten ključ. Za pravilen prikaz elementov pri izbiranju posamezne občine pa poskrbimo tako, da vsakemu elementu podamo še unikaten ključ za občino, iz katere prihaja ljudska pesem.



Slika 13: Izgled implementacije vizualizacije zvočnih zbirk

Poglavje 4 Sklepne ugotovitve

Zaključimo lahko z opazko, da je vizualizacija podatkov v računalništvu zanimiva tema, ki bo verjetno deležna še burnega razvoja, glede na to, da se uporablja na virtualno vseh področjih. Verjetno bo dobila še dodatno težo (pomen, mesto) v umetnosti in poučevanju. Trenutno ji manjkajo dobre konvencije in nek večji, po možnosti odprtokodni projekt, ki bi integriral vso tehnologijo za izdelavo v neko skupno platformo. D3.js to sicer deloma že rešuje. Programerju bi svetovali, da v kolikor želi hitro ustvariti kakšno vizualizacijo, naj to naredi s tehnologijami, ki jih pozna, in izbere tak tip vizualizacije, ki bo enostaven za izdelavo. Če pa želi izdelati bolj kompleksen ali čisto nov tip, potem je tehnologija D3.js vsekakor dobro izhodišče. V implementaciji primera se je dobro izkazala, saj nam je olajšala nalaganje podatkov iz zbirke pesmi in njihovo vizualizacijo. Mogoče še dodaten namig: če želimo pritegniti širše občinstvo k interakciji z vizualizacijo, ki ne prikazuje nekih pomembnih podatkov, potem se lahko bolj posvetimo izgledu, da jih bomo impresionirali in posledično bolj pritegnili k interakciji (tak primer je vizualizacija Streamchart). Na koncu bi želeli tudi spodbuditi računalničarje in druge ljudi, ki se jim zdi vizualizacija podatkov zanimiva, da se v tej smeri izobražujejo, saj ima panoga perspektivo. Če ne drugega, bodo lahko pri njeni implementaciji in interakciji spoznali adaptivne procese, ki jih naši možgani izvajajo že ves čas, pa na to nismo bili nikoli pozorni.

Literatura

[1] "Using Visualization to share the human impact of numbers", 2014, dostopno na: http://www.datafactz.com/blog/2014/09/19/using-visualization-to-share-the-human-impact-of-numbers/

[2] Noah Iliinsky, Julie Steele, "Designing Data Visualizations", O'Reilly Media, 2011, str. 3-11

[3] Edward R. Tufte, "The Visual Display of Quantitative Information", Graphics Pr, 2001

[4] Wikipedia, "Data visualization", dostopno na: http://en.wikipedia.org/wiki/Data_visualization

[5] Martin Halvey, Mark T. Keane, "An Assessment of Tag Presentation Techniques", IW3C2, 2007, dostopno na: http://www2007.org/htmlposters/poster988/

[6] Michael Burch, Steffen Lohmann, Daniel Pompe, Daniel Weiskopf, "Prefix Tag Clouds", 2013, dostopno na: http://www.vis.uni-stuttgart.de/uploads/tx_vispublications/PrefixTagClouds-IV2013.pdf

[7] "HiveR: 2D and 3D Hive Plots for R", dostopno na: http://cran.r-project.org/web/packages/HiveR/index.html

[8] "Present your data in a bubble chart", dostopno na:http://office.microsoft.com/en-001/excel-help/present-your-data-in-a-bubble-chart-HA001233749.aspx

[9] Ben Shneiderman, "Treemaps for space-constrained visualization of hierarchies", 1998, dostopno na: http://www.cs.umd.edu/hcil/treemap-history/index.shtml

[10] Peter Dizikes, "Better product design through a simple square chart", MIT News Office, 2012, dostopno na: http://newsoffice.mit.edu/2012/design-structure-matrix-modeling-0730

[11] Lee Byron, Martin Wattenberg, "Stacked Graphs – Geometry & Aesthetics", 2008, dostopno na: http://hint.fm/papers/leebyron_stackedgraphs_byron_wattenberg.pdf

[12] "Streamgraph: Microsoft Research Data Visualization Apps for Office", dostopno na: http://research.microsoft.com/en-us/projects/msrdatavis/streamgraph.aspx

[13] Michael Bostock, Vadim Ogievetsky, Jeffrey Heer, "D3: Data-Driven Documents", 2011, dostopno na: http://vis.stanford.edu/files/2011-D3-InfoVis.pdf

[14] Scott Murray, "Interactive Data Visualization for the Web", O'Reilly Media, 2013, str. 7-10

[15] "D3 API Reference", dostopno na: https://github.com/mbostock/d3/wiki/API-Reference

[16] Tim Bray, "The JavaScript Object Notation (JSON) Data Interchange Format", Internet Engineering Task Force (IETF), 2014, dostopno na: http://tools.ietf.org/html/rfc7159

[17] Ecma International, "The JSON Data Interchange Format(ECMA-404)", dostopno na: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[18] "The GeoJSON Format Specification", dostopno na: http://geojson.org/geojson-spec.html

[19] "Scalable Vector Graphics (SVG) 2", W3C Editor's Draft 10 October 2014, dostopno na: https://svgwg.org/svg2-draft/intro.html

[20] C. Concolato, J.-C. Dufourd, "Comparison of mpeg-4 bifs and some other multimedia description languages", 2002, dostopno na: http://perso.telecom-paristech.fr/~concolat/Papers/2002%20-%20WEMP%20-%20Comparison%20BIFS%20and%20others.pdf

[21] Wikipedia, "SWF", dostopno na: http://en.wikipedia.org/wiki/SWF

[22] Wikipedia, "Adobe Flash Player", dostopno na: http://en.wikipedia.org/wiki/Adobe_Flash_Player

[23] Wikipedia, "Adobe Flash", http://en.wikipedia.org/wiki/Adobe_Flash

[24] Ecma International, "ECMAScript for XML (E4X) Specification(ECMA-357)", dostopno na: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-357.pdf

[25] "Flare tutorial, Introduction, Part 3: Visualizations", dostopno na: http://flare.prefuse.org/tutorial

[26] "Microsoft Application Architecture Guide, 2nd Edition - Chapter 23: Designing Rich Internet Applications", Microsoft Press, 2009, dostopno na: http://msdn.microsoft.com/en-us/library/ee658083.aspx

[27] Michael Friendly, "Milestones in the history of thematic cartography, statistical graphics, and data visualization", 2009, str. 3-39, dostopno na: http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf