Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Vizualizacija podataka

Seminarski rad

Vizualizacija prodaja video igara (2019.)

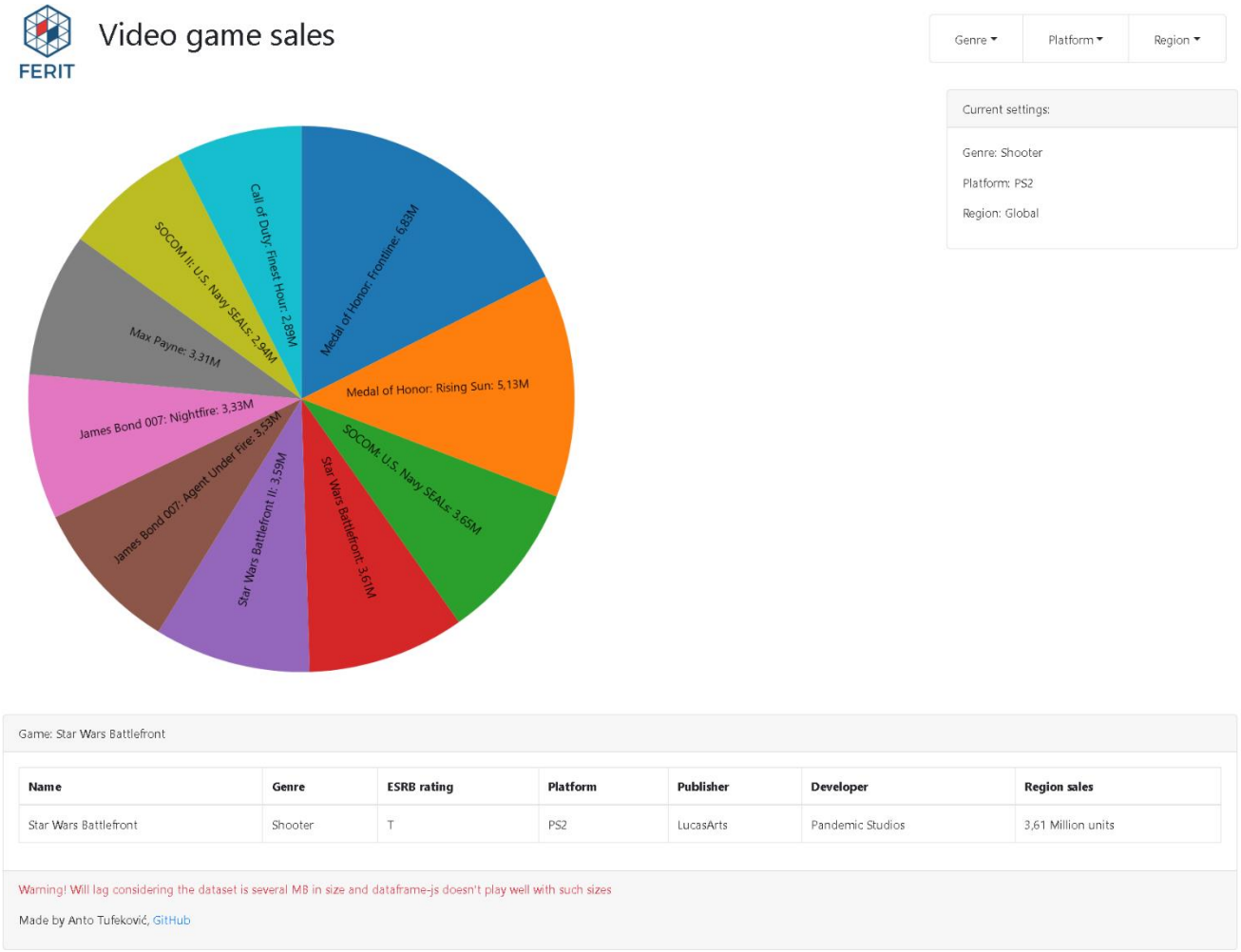
Anto Tufeković

Osijek, 2020.

Sadržaj

1. Uvod

U sklopu projekta je napravljena web stranica koja grafički prikazuje udio prvih 10 (ili manje ako nema dovoljno unosa) video igara zajedno sa izbornicima za odabir vrste podataka za pregled (npr. može se promijeniti regija sa Sjeverne Amerike na Japan ili žanr sa puzli na avanture). Podatci prikazani su u milijunima prodanih jedinica, te nije moguće izvući neku valutu iz ovih podataka jer moguće je da su se prodavali npr. pod rasprodajom. Klikom na nekog člana grafa će prikazati dodatne informacije o igri u tablici na web stranici (Slika 1.1).



Slika 1.1 – prikaz web stranice sa kliknutom igrom za više informacija

Web stranica je implementirana koristeći JavaScript kao podlogu, dataframe-js biblioteku[1] za učitavanje podataka i lakše traženje i d3 biblioteku[2] koja je korištena za generiranje grafičkog prikaza. Za razmještaj elemenata po stranici je korišten Bootstrap 4[3]. Python i Microsoft Office Excel su korišteni za čišćenje podataka.

2. Korišten skup podataka

Korištena je stranica Kaggle za odabir baze podataka te iz nje smo povukli bazu „Video Games Sales 2019“ koju je kreirao Abdulshaheed Alqunber[4]. Baza sadrži više stupaca za svaki redak, gdje redci predstavljaju igre dok stupci predstavljaju informacije o igrama.

Iz opisa baze podataka:

„Postoje 55,792 unosa u bazu podataka od 12.4.2019., generirano je „struganjem“ web stranice vgchartz.com“

Svaki unos sadrži sveopći rang, ime, platformu, žanr, ESRB procjenu, izdavača, razvijača(developer), ocjenu kritika(ako je dostupno), ocjenu korisnika(ako je dostupno), sveukupno prodanih jedinica, prodane jedinica za Sjevernu Ameriku, Europu, Japan i ostalo, te godinu izdanja igre.

Na ovaj način se može prosuditi po regiji, žanru i platformi kako idu prodaje igara, što se koristi u ovom radu.

Zbog prevelike količine informacija se ne koriste prodaje ostalih zemalja, ocjena kritika i ocjena korisnika.

2.1. CSV(Comma Separated Value)

U ovom obliku dolazi baza podataka. CSV ili „zarezom odvojene vrijednosti“ se pojavljuje oko 1978. godine[5] u Fortran-u u obliku slobodnih popisa gdje svaki red prestavlja jedan unos i vrijednosti su odvojene zarezom ili razmakom, te same vrijednosti ne smiju imati zarez ili razmak u sebi. CSV se pojavljuje kao ime i skraćenica u Osborne računalima koji koriste SuperCalc proračunske tablice.

Formalno je definirana 2005. kao MIME tip sadržaja za spremanje podataka te uvedena je opća podrška za raznim vrstama različitih graničnika(engl. *delimiter*) ovisno o potrebi (npr. proračunske tablice za financije koriste i točke i zareze pa se može koristiti npr. točka-zarez ili neki sličan nekorišten simbol).

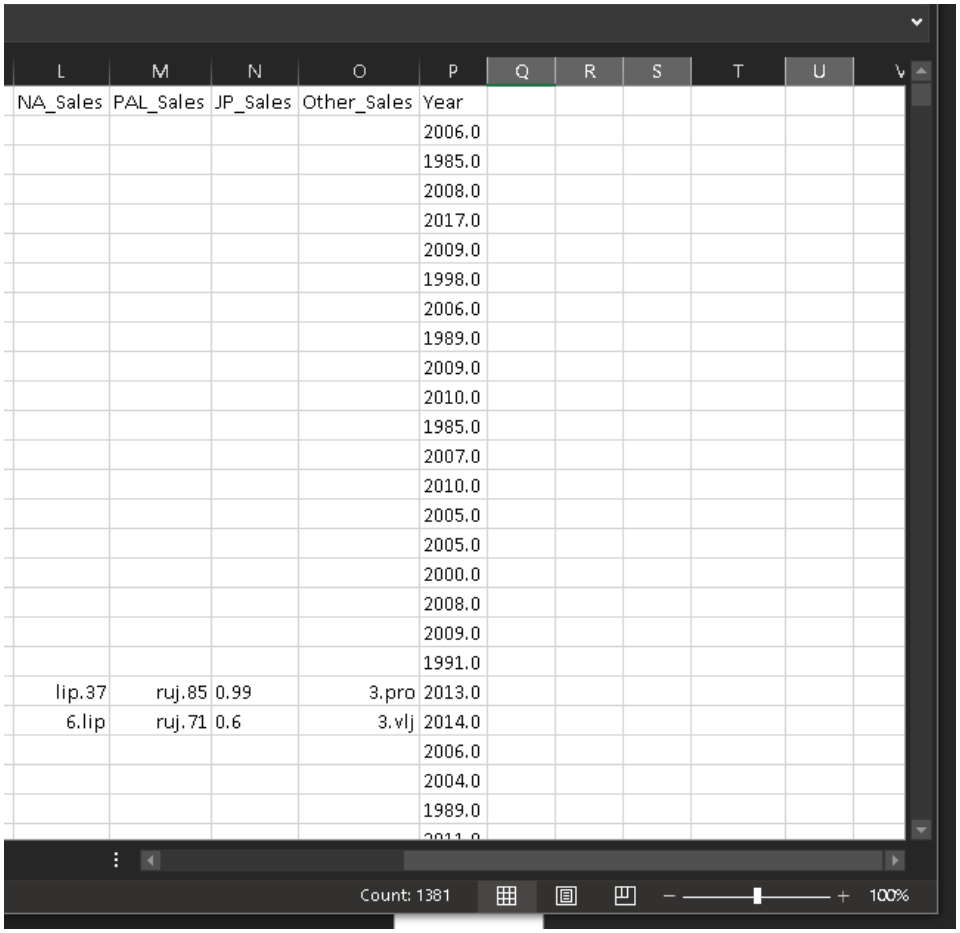
2.2. Prilagodba skupa podataka

Skup podataka skinut sa interneta dolazi onečišćen, što znači da se treba prvo očistiti prije korištenja inače dolazi do problema. Korištenjem python koda saznajemo da postoje redci sa više stupaca nego drugi.

```
import numpy as np
import pandas as pd

csv = pd.read_csv("./smol.csv", ";")
print("CSV file has " + str(len(csv.index)) + " rows")
print(csv.head())
```

Ovaj kod nam daje grešku da nema postojane količine stupaca u .csv datoteci. Učitavanjem .csv datoteke u Excel te odabiranjem redaka sa podatcima u stupcima iznad traženih daje nam sljedeće:



L	M	N	O	P	Q	R	S	T	U
NA_Sales	PAL_Sales	JP_Sales	Other_Sales	Year					
				2006.0					
				1985.0					
				2008.0					
				2017.0					
				2009.0					
				1998.0					
				2006.0					
				1989.0					
				2009.0					
				2010.0					
				1985.0					
				2007.0					
				2010.0					
				2005.0					
				2005.0					
				2000.0					
				2008.0					
				2009.0					
				1991.0					
lip.37	ruj.85	0.99	3.pro	2013.0					
6.lip	ruj.71	0.6	3.vlj	2014.0					
				2006.0					
				2004.0					
				1989.0					

Slika 2.1 – dokaz prljavih podataka

Vidimo 1381 unosa koji nisu pravilno formatirani, tj. sadrže u sebi zarez bez navodnika te to ometa programima te bacaju greške zbog toga. Previše je podataka da se rukom poprave te su navedeni retci obrisani. Nakon toga smo Total_Shipped stupac stopili sa Global_Sales jer su ista stvar (gdje nema prvog stupca ima drugog i suprotno, lakše je raditi tako da se stope zajedno te znače istu stvar, ostavljeno je ovako jer je originalno došlo na ovaj način sa struganja stranice). Konačno spremamo datoteku te imamo čistu datoteku za korištenje. Sada nam prethodni python kod daje pravilno izlistanje podataka te možemo ga dalje koristiti za naše potrebe.

3. Implementacija

3.1. Učitavanje podataka

Sljedeći blok koda nam služi za učitavanje podataka:

```
//ucitaj podatke koristeci dataframe-js biblioteku
DataFrame.fromCSV("src/smol.csv").then(df => {
    this.data = df.select('Name', 'Genre', 'ESRB_Rating', 'Platform', 'Publisher',
 'Developer', 'Global_Sales', 'NA_Sales', 'PAL_Sales', 'JP_Sales', 'Year');
...
```

U navedenom kodu koristimo ugrađenu funkciju *fromCSV* da bi učitali podatke iz .csv datoteke. Ovo je lokalno izvedeno pa je relativno brzo ali na internetu bi trebalo prvo prebaciti .csv datoteku koja je skoro 5MB u veličini, ali datoteka ostaje spremljena u cache spremištu preglednika tako da nije potrebno ponovno učitavanje ako se stranica osvježi. Najviše resursa zahtjeva korištenje *df.select* i *df.filter* funkcija kojom odabiremo tj. filtriramo tražene podatke po uvjetima. One uvelike usporavaju rad preglednika te zbog toga treba par sekundi procesiranja da bi se dobili potrebni podatci.

3.2. Priprema elemenata

Nakon što su podatci dobiveni dohvaćaju se svi HTML elementi:

```
...
this.setEvents();

let settingGenre = document.querySelector("#textGenre");
let settingPlatform = document.querySelector("#textPlatform");
let settingRegion = document.querySelector("#textRegion");
this.settingInfoText.push(settingGenre, settingPlatform, settingRegion);
this.spinner = document.querySelector("#loaderSpinner");
this.svgContainer = document.querySelector("#svgContainer");
this.tableClicked = document.querySelector("#tableClicked");
this.textClickedGameTitle = document.querySelector("#textClickedGameTitle");
if (this.verbose) { console.log(this.settingInfoText); }

//na kraju se prikaz osvježava
this.updateDisplay();
});
```

Nakon što su dohvaćeni svi elementi može se nastaviti sa ostalom logikom koda.

Funkcija *this.setEvents()* nam služi za postavljanje događaja kada korisnik klikne na tipke u izborniku za mijenjanje žanra, platforme ili regije.

```
setEvents() {
    this.genres.forEach(genre => {
        let temp = document.querySelector("#button" + genre);
        this.eventClickHandlerChange(temp, "genre", genre);
        this.genreButtons.push(temp);
```

```
        });
    if (this.verbose) { console.log(this.genreButtons); }
    this.platforms.forEach(platform => {
        let temp = document.querySelector("#button" + platform);
        this.eventClickHandlerChange(temp, "platform", platform);
        this.platformButtons.push(temp);
    });
    if (this.verbose) { console.log(this.platformButtons); }
    this.regions.forEach(region => {
        let temp = document.querySelector("#button" + region);
        this.eventClickHandlerChange(temp, "region", region);
        this.regionButtons.push(temp);
    });
    if (this.verbose) { console.log(this.regionButtons); }
}
```

```
eventClickHandlerChange(button, type, value) {
    if (type == "genre") {
        button.addEventListener("click", (e) => {
            this.selectedGenre = value;
            this.updateDisplay();
        });
    } else if (type == "platform") {
        button.addEventListener("click", (e) => {
            this.selectedPlatform = value;
            this.updateDisplay();
        });
    } else if (type == "region") {
        button.addEventListener("click", (e) => {
            this.selectedRegion = value;
            this.updateDisplay();
        });
    } else {
         console.log("Something went wrong with setting on click events with follo
wing parameters (button, type, value):");
         console.log(button);
         console.log(type);
         console.log(value);
    }
}
```

Svaka tipka je dohvaćena i odvojena po izborniku, pa se u funkciji *this.eventClickHandlerChange()* postavlja funkcija tipke ovisno o vrijednosti tipke (npr. tipka buttonMusic će imati vrijednost Music).

3.3. Postaljanje elemenata i prikaza

U funkciji *this.updateDisplay()* se nalazi potreban kod za osvježavanje sadržaja stranice:

```
updateDisplay() {
     if (this.verbose) {
        console.log(this.selectedGenre);
        console.log(this.selectedPlatform);
        console.log(this.selectedRegion);
    }
    this.settingInfoText[0].innerHTML = "Genre: " + this.selectedGenre;
    this.settingInfoText[1].innerHTML = "Platform: " + this.selectedPlatform;
    this.settingInfoText[2].innerHTML = "Region: " + this.selectedRegion;
    let dataArray = this.getTopTenFromData();
    this.showGraph(dataArray);
}
```

U dohvaćenim HTML elementima postavljamo pripadajuće vrijednosti kad se mijenja izbor korisnika. Funkcija *this.getTopTenFromData()* nam daje prvih deset redova iz baze podataka ovisno o izboru korisnika:

```
getTopTenFromData() {
    let temp = this.data.select('Name', 'Genre', 'ESRB_Rating', 'Platform', 'Publi
sher', 'Developer', this.selectedRegion + '_Sales', 'Year').filter({ 'Genre': this
.selectedGenre, 'Platform': this.selectedPlatform }).filter(row => row.get(this.se
lectedRegion + '_Sales') != "");
    if (this.verbose) {
        console.log("Gotten " + temp.count() + " row(s) of data");
        console.log(temp.head(10).toArray());
    }
    return temp.head(10).toArray();
}
```

Tu se koriste prethodno spomenute funkcije koje nisu optimizirane za velike količine podataka *select()* i *filter()*. Funkcija vraća prvih deset te pretvara ih usput u polje koje je čitljivo d3 funkcijama.

U funkciji *this.showGraph()* se obavlja crtanje grafa:

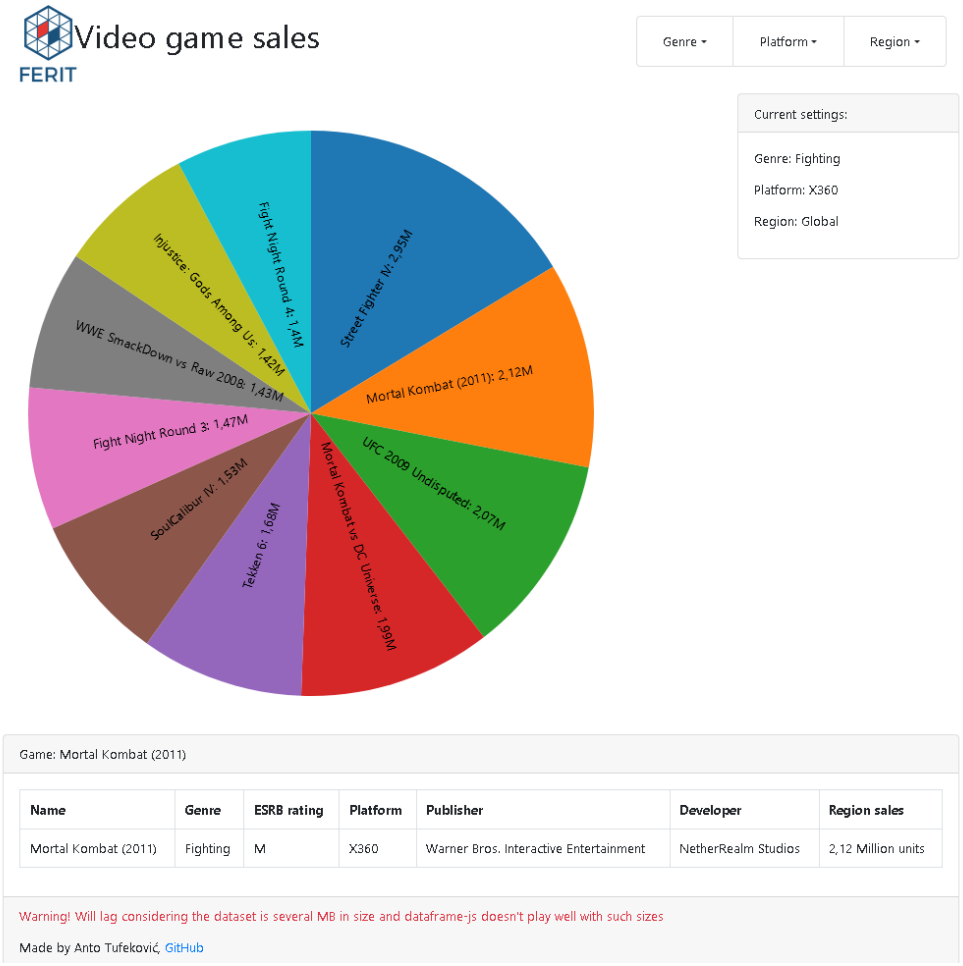
```
showGraph(dataArray) {
    if (this.svgContainer.hasChildNodes()) {
        this.svgContainer.innerHTML = "";
    }
    let width = 800;
    let height = 800;
    let outerRadius = (Math.min(width, height) / 2) * 0.9;
    let innerRadius = 0;
    ...
    pieArcs.on("click", (d) => {
```

```
        if (this.verbose) { console.log(d.data); }
         let temp = "<td>" + d.data[0] + "</td>" + "<td>" + d.data[1] + "</td>" +
"<td>" + d.data[2] + "</td>" + "<td>" + d.data[3] + "</td>" + "<td>" + d.data[4] +
 "</td>" + "<td>" + d.data[5] + "</td>" + "<td>" + d.data[6] + " Million units</td
>";
         this.tableClicked.rows[1].innerHTML = temp;
         this.textClickedGameTitle.innerHTML = "<div id='loaderSpinner'></div>Game
: " + d.data[0];
         });
}
```

Funkcija prvo provjerava dali kontejner za SVG element ima neke članove, ako ima obriši ih. Nakon toga nastavlja sa crtanjem kružnog grafa za prikaz prodaja (identično kao sa laboratorijskih vježbi). U ovom dijelu koda bi se namještala veličina grafa. Na kraju se svakom isječku kružnog grafa pridjeljuje događaj na klik gdje će se opširniji podatci o igri ispisati u tablici koja se nalazi ispod kružnog grafa.

3.4. Konačan rezultat

Na kraju se dobije spor ali funkcionalan grafički prikaz sa dodatnim podatcima na klik.



Slika 3.1 – prikaz potpunog odabira

U slučaju nedostatka podataka (gdje vrijednost stupca ne postoji) jednostavno se ispišu kojigod podatci postoje, npr. odabirom Shooter, Nintendo Switch, Japan dobijemo:

Slika 3.2 – Prikaz kada nedostaju podatci

Vidimo da postoje samo dva retka sa postojećim podatcima. Ako prebacimo natrag na Global dobijemo:

Slika 3.3 – alternativni prikaz sa odabranom globalnom regijom

Ovdje vidimo puno realističnije podatke jer postoje više unosa, ovo je zbog toga što na stranici sa koje su podatci dobiveni (vgchartz.com) nije zapisano za npr. Splatoon prodaje u Japanu, iako je igra nastala u Japanu.

4. Zaključak

Prodaje video igara se stalno prate jer po tome se zaključuje dali je igra bila uspješna ili ne, tako da koristeći usporedbu najboljih ≤ 10 igara daje dobru predodžbu koliko je jedna igra bila uspješnija naspram drugih.

Ova ideja je bila inspirirana sličnim stranicama sa interneta kao što je steamspy.com ili steamdb.info.

Moguća poboljšanja stranici bila bi podrška za mobilne uređaje, to bi se moglo izvesti kreiranjem zasebnog rasporeda elemenata za mobilne korisnike te korištenjem prilagođenog grafičkog prikaza, jer trenutno je stalne veličine 800x800, što nije pogodno za mobilne stranice. Još jedno moguće poboljšanje je pronaći rješenje za problem sporosti koja donosi biblioteka dataframe-js, npr. spremivši .csv datoteku u dvodimenzionalno polje pa pisati prilagođene funkcije za brže sortiranje i filtriranje.

Literatura

[1] Guillaume Mousnier, Gmousse/dataframe-js, https://gmousse.gitbooks.io/dataframe-js/

[2] Mike Bostock, Data-Driven Documents (d3), https://github.com/d3/d3

[3] Bootstrap Team, Bootstrap, https://getbootstrap.com/

[4] Abdulshaheed Alqunber, Video Games Sales 2019, https://www.kaggle.com/ashaheedq/video-games-sales-2019

[5] Nepoznato, Comma-separated values, https://en.wikipedia.org/wiki/Comma-separated_values

Dostupnost rješenja

Github link: https://github.com/ATufekovic/VP_projekt