VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VIZUALIZACE DAT.

Alois Kužela

BRNO 2008

Obsah

Úvod....3
1 Vizualizace dat....4
1.1 Sedm fází procesu vizualizace....5
1.1.1 Získání dat (acquire)....5
1.1.2 Parsování dat (parse)....5
1.1.3 Filtrace (filter)....5
1.1.4 Dolování (mine)....5
1.1.5 Reprezentace (represent)....6
1.1.6 Vylepšení (refine)....6
1.1.7 Interakce (interact)....6
1.2 Detailnější příklad....6
1.2.1 Získání dat....6
1.2.2 Parsování....7
1.2.3 Filtrace....7
1.2.4 Dolování....7
1.2.5 Reprezentace....8
1.2.6 Vylepšení....9
1.2.7 Interakce....9
1.3 Shrnutí....11
2 Představení aplikace Processing....12
3 Mapa USA....15
3.1 Zobrazení dalších informací....19
3.2 Souhrn....20
4 Za všechno může čas ….....21
5 Různé příklady zobrazení dat....25
6 Závěr....34
Reference....35
Příloha....36

Úvod

Tento text je spíše referátem ke knize Visuaizing Data [1]. Popíše nejdůležitější myšlenky sesbírané z jednotlivých kapitol této knihy a přiblíží tak podstatu probírané tématiky. Kniha je psána stylem ukázky rozboru zadaného příkladu, kdy se autor krok za krokem dobírá k řešení a vytvoření aplikace prezentující data. Postupně se dostává ke stále složitějším aplikacím s odpovídajícím nárůstem zdrojových kódů s tím, že základní postupy a myšlenky se opakují, a mění se pouze metody, kterými jsou řešeny.

Tato kniha představuje čtenářům jeden z často řešených problémy poslední doby – jakým způsobem zpracovat velké množství dat a jak je vhodně prezentovat.

V několika kapitolách předvádí základní principy, myšlenky a postupy autora při vizualizaci dat na konkrétních příkladech, díky čemuž je autor schopen přiblížit toto téma širokému spektru čtenářů. Kniha je koncipována tak, že se postupně předkládají stále složitější problémy a je předvedeno jejich řešení. V pozdějších kapitolách je tedy již nutné mít osvojeny alespoň základy programování pro pochopení uvedených algoritmů řešení. Na druhou stranu každý krok vedoucí k vyřešení konkrétního úkolu je dostatečně okomentován, takže si z kniha je přínosem jak pro začátečníky (neprogramátory) tak pro znalejší čtenáře.

Základem knihy, na kterém je celý obsah vystavěn, je autorem vyvinutý nástroj či jednoduché programovací prostředí nazvané *Processing*. Toto prostředí je založeno na programovacím jazyku Java a pomocí vhodně zvolené syntaxe a zapouzdření obecně složitějších úkonů pro práci s grafikou či soubory do jednoduchého volání funkcí je dle mého názoru vynikajícím nástrojem pro výuku programování i pro tvorbu aplikací pro vizualizaci dat. I bez znalosti jazyka Java si uživatel snadno osvojí použitou syntaxi a princip tvorby programu a je schopen poměrně snadno a rychle vytvořit aplikaci reprezentující data na opravdu dobré úrovní – tabulky, grafy, stromy, interakce či interpolace jednotlivých hodnot apod.

Osobně bych doporučil tento nástroj přinejmenším vyzkoušet (www.processing.org) a zvážit jeho začlenění do osnov některého z vyučovaných kurzů.

1 Vizualizace dat

Topíme se v datech. To je přímý důsledek moderního rozvoje naší společnosti. Především díky stále výkonnějším a zároveň levnějším počítačům a zařízením pro ukládání dat lidé mají neustále větší tendence shromažďovat čím dál větší objemy dat. S rozmachem internetu, kde lze najít v podstatě *cokoliv* se lidé naučili často vyhledávat a shromažďovat nejrůznější informace pro *pozdější použití*. Ovšem využití nashromážděných dat po té již pokulhává a stávají se tak zbytečně zastaralými a nevyužitými.

Pokud se rozhodneme zpracovávat nějaká data a předvést vhodným způsobem získané informace, musíme si být vědomi několika skutečností. Dvě nejdůležitější jsou:

1. data se neustále mění, každým okamžikem se něco děje a informace vznikají a zastarávají. Jako příklad autor uvádí knihu *Information Anxiety* od R. S. Wurmana podle níž nedělní vydání novin New York Times obsahuje více informací než ke kolika měl za celý svůj přístup člověk v období renezance. Neskutečné množství.
Pokud chceme vyvinout aplikaci zachycující některý rys, je nutné rozmyslet si, zda se bude jednat pouze zachycení *historického vývoje* do určitého časového okamžiku nebo jakým způsobem zajistíme aktuálnost zobrazených informací.
2. Jak vhodně zobrazit obrovskou kolekci dat tak, aby byla přehledná a zejména snadno pochopitelná pro ostatní? Existuje několik často používaných způsobů zobrazení dat:
   - tabulky,
   - grafy – nejrůznější druhy,
   - stromy pro hierarchické struktury,
   - sítě,
   - schema,
   - a další.

   Je důležité umět vybrat správnou reprezentaci a zajistit její srozumitelnost. Jako krásný příklad jsou uváděny navigační mapy v metrech, vymyšlené H. Beckem ve třicátých letech, kdy je kompletně abstrahován skutečný tvar trasy metra (na rozdíl od mapy silnic) a pomocí horizontálních, vertikálních a diagonálních čar je zachyceno pouze to podstatné v přehledné a snadno srozumitelné formě: Trasy, stanice na které se lze jednotlivými trasami dostat, případně kde se trasy kříží a lze přesednout na potřebnou trasu.
   Je důležité si tedy i uvědomit, co člověk v daném okamžiku chce či potřebuje znát a podle toho přizpůsobit danou reprezentaci a úroveň zobrazených dat. V uvedeném případě potřebuje pouze zodpovědět otázku: „Jak se dostanu z místa A do místa B?“.

V cílové stanici pak nalezne mapu části daného města, podle které se může již detailně zorientovat a nalézt potřebný směr pro jeho další cestu. Stejně tak by měla aplikace poskytovat i z počátku skryté informace formou interakce či zvětšením detailu zobrazení.

Na několika příkladech bude dále ukázáno jakým způsobem je vhodné postupovat při návrhu a postupném vylepšování aplikace. Zde uvedu pouze jednu velice jednoduchou, ale velmi silnou pomůcku: **Nepřehánět to s množstvím zobrazených dat. Vybrat co nejmenší množství dat, které vyjádří podstatu**.

Vizualizace dat je jednou z fází dolování dat, proto je kombinací více disciplín jako matematiky (zejména statistika), dolování dat, grafika (pro vlastní zobrazení dat) atd. Celý proces návrhu lze rozdělit do sedmi hlavních kroků, díky čemuž lze např. snadno spolupracovat s odborníky z daného oboru pro každou fázi vývoje.

1.1 Sedm fází procesu vizualizace

1.1.1 Získání dat (acquire)

Získání potřebných dat, podkladů, materiálů. Můžeme zpracovávat data z pevného disku či přímo ze zdroje na síti či internetu.

1.1.2 Parsování dat (parse)

Ve většině případů nejsou podklady, které máme k dispozici v takovém tvaru, který můžeme ihned použít. Je nutno vytvořit či rozpoznat strukturu dat a zařadit data do jednotlivých kategorií.

Často se spojuje s fází filtrace, kdy se rovnou relevantní podmnožina dat z podkladů převede do tvaru vhodného pro naši vytvářenou aplikaci (xml, csv, tsv formát).

1.1.3 Filtrace (filter)

Odstranit všechna nepotřebná data (čištění dat, odstranit vše co pro nás momentálně nemá žádný přínos).

1.1.4 Dolování (mine)

Použití statistických metod či dalších metod z oblasti dolování dat pro rozpoznání vzorů či vytvoření matematického popisu dat.

1.1.5 Reprezentace (represent)

Výběr vhodného základního vizuálního modelu – graf, seznam, strom, …

1.1.6 Vylepšení (refine)

Vylepšit základní reprezentaci dat, aby byla jasnější a vizuálně přívětivější.

1.1.7 Interakce (interact)

Přidat metody pro manipulaci dat – ovládání zobrazovaných částí.

Toto jsou základní části, na které lze rozdělit celý proces vytváření vhodné vizualizace dat. Ovšem závisí vždy na okolnostech kolik kroků bude použito. Ve většině případů lze více kroků sloučit do jednoho (získání, parsování a filtrování dat), případně některé kroky vynechat (ne vždy potřebujeme vyhledávat nějaké skryté vzory v datech – vynecháme fázi dolování).
Následuje detailnější příklad s popisem všech důležitých myšlenek při tvorbě základního modelu a dále jeho vylepšování. V ostatních příkladech budou uvedeny vesměs pouze obrázky a nejdůležitější důvody pro výběr použité reprezentace. Zájemce o podrobnější vysvětlení celého procesu odkazuji na prostudování samotné publikace.

1.2 Detailnější příklad

Zajímá nás rozdělení ZIP (Zone Improvement Plan, obdoba našeho PSČ) napříč USA.

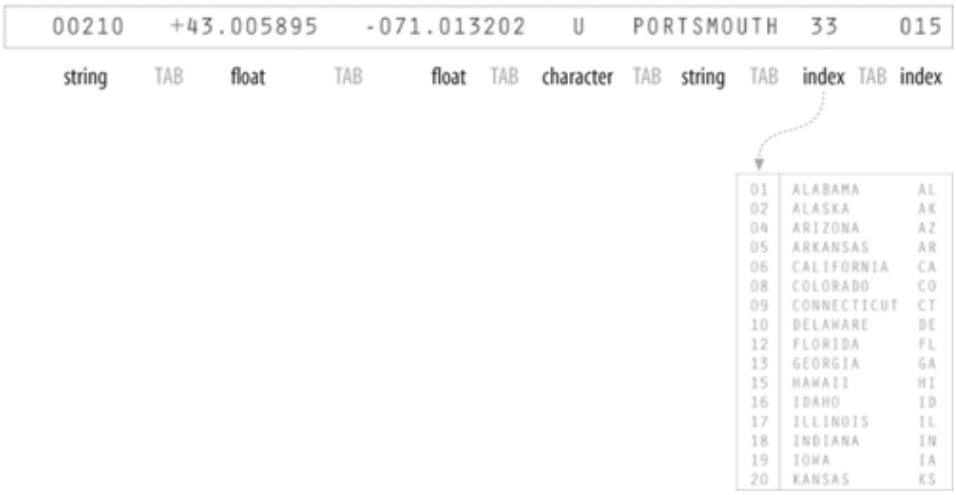
1.2.1 Získání dat

Vhodným místem pro hledaní takového typu dat jsou webové stránky vládních orgánů. Pro tento příklad byla použita data získána ze stránek U.S. Census Bureau. Získaný soubor obsahuje zhruba 42 000 řádků ve tvaru:

00210	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00211	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00212	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00213	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00214	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00215	+43.005895	-071.013202	U	PORTSMOUTH	33	015
00501	+40.922326	-072.637078	U	HOLTSVILLE	36	103
00544	+40.922326	-072.637078	U	HOLTSVILLE	36	103
00601	+18.165273	-066.722583		ADJUNTAS	72	001
00602	+18.393103	-067.180953		AGUADA	72	003
00603	+18.455913	-067.145780		AGUADILLA	72	005
00604	+18.493520	-067.135883		AGUADILLA	72	005
00605	+18.465162	-067.141486	P	AGUADILLA	72	005
00606	+18.172947	-066.944111		MARICAO	72	093
00610	+18.288685	-067.139696		ANASCO	72	011
00611	+18.279531	-066.802170	P	ANGELES	72	141
00612	+18.450674	-066.698262		ARECIBO	72	013
00613	+18.458093	-066.732732	P	ARECIBO	72	013
00614	+18.429675	-066.674506	P	ARECIBO	72	013
00616	+18.444792	-066.640678		BAJADERO	72	013

1.2.2 Parsování

Nyní je důležité rozpoznat strukturu a význam jednotlivých záznamů v souboru, abychom mohli vytvořit algoritmus pro výběr podstatných částí. Pro nás bude zajímavá zeměpisná šířka a délka, název města, ZIP kód a index státu. Obecná struktura získaného souboru je uvedena na následujícím obrázku ve tvaru vhodném pro snadné algoritmické zpracování:



Obrázek 1.2. Struktura použitého souboru [1].

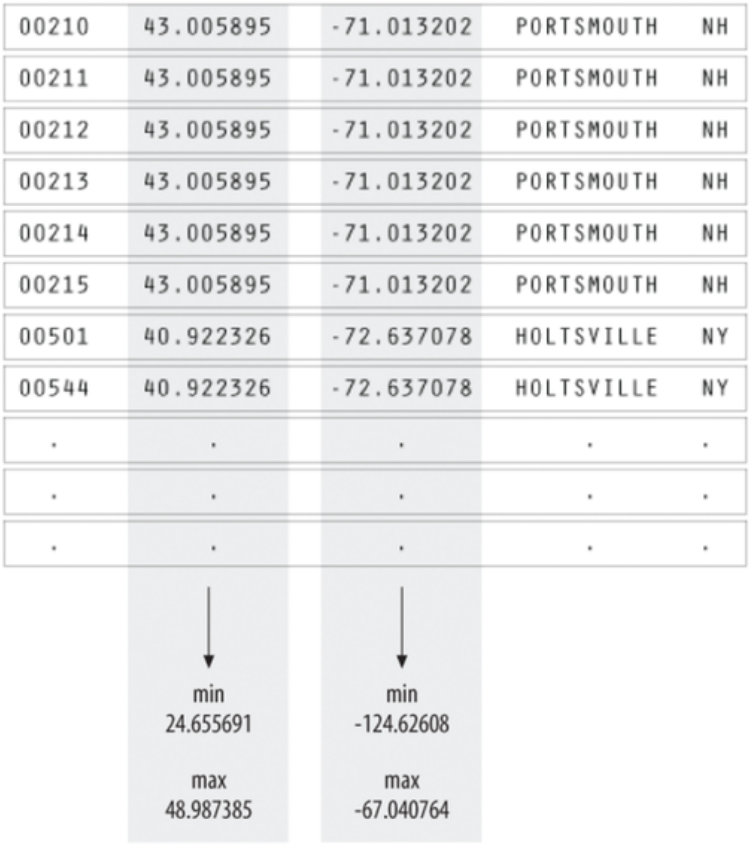
1.2.3 Filtrace

Zajímavé pro nás budou pouze ZIP kódy hlavních 48 států. Proto je možné *zahodit* všechny řádky s údaji pro Aljašku, Havaj a podobně.

K propojení indexů států a rozpoznání, které záznamy lze zanedbat, potřebujeme další soubor mapující kódy na názvy a zkratky států (naznačeno tabulkou na obr. 1.2)

1.2.4 Dolování

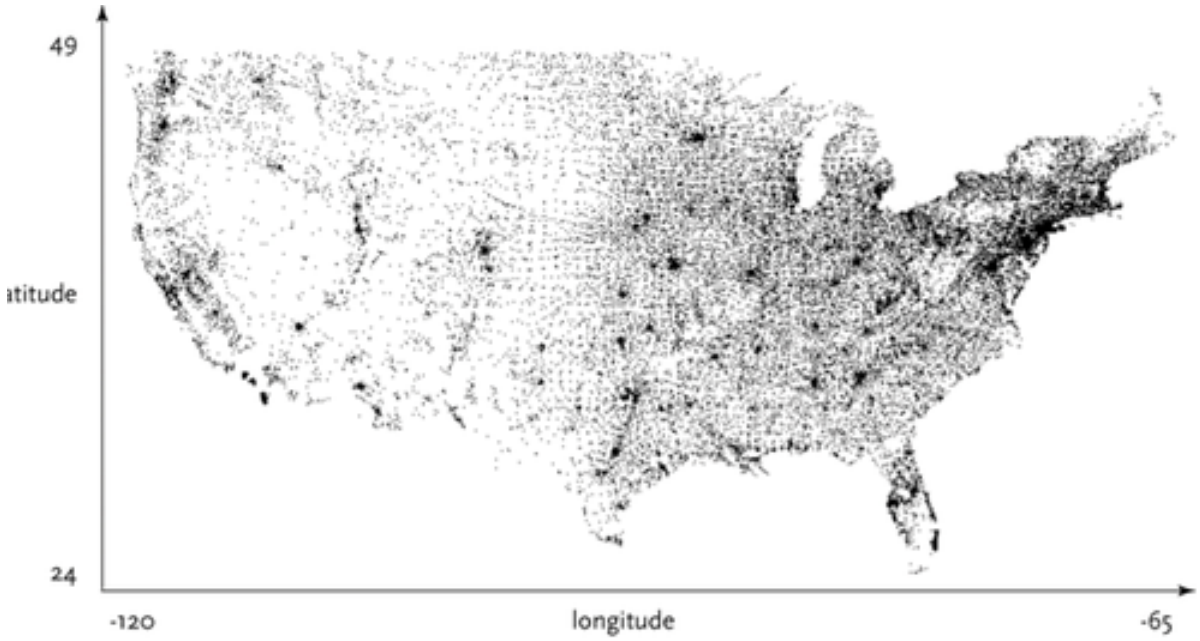
V tomto příkladu nám postačí zjistit minimální a maximální hodnoty pro zeměpisnou šířku a délku pro zajištění vhodného mapování údajů na zobrazovací plochu. Toto lze získat jednoduchým průchodem všemi daty (vhodné spojit s předchozími fázemi).



Obrázek 1.3. Tvar výsledného vstupního souboru s výsledkem dolování (min, max) [1].

1.2.5 Reprezentace

V této chvíli je nutno zvolit jakým způsobem zobrazit získaná data. Některá z nich měla tvar seznamu, jiná byla stromové struktury (mapování indexů na názvy států) apod. Jelikož každý ZIP kód má přiřazenu zeměpisnou polohu, je vhodné jí využít jako souřadnice dvojrozměrného grafu. Výsledek je zobrazen na obrázku 1.4., kde díky velkému množství vstupních dat získáme téměř věrohodnou mapu měst USA. Toto řešení je ale nedostačující, protože původní otázka ohledně ZIP kódů zůstává nezodpovězena.



Obrázek 1.4. Výsledek základní reprezentace [1].

1.2.6 Vylepšení

V takovém rozsáhlém množství dat je nereálné tímto způsobem prezentovat všechny hledané informace – ZIP kódy jednotlivých míst. I při zobrazení pouhých bodových souřadnic vznikají shluky překrývajících se oblastí a texty by tudíž byly zcela nečitelné. Vhodným řešením je zapojit interakci s uživatelem. Pro její použití stačí ve vizualizaci změnit barvy – pozadí podkladu na černou, body měst žlutou a oblasti s vyhovujícím ZIP kódem bílou barvou. Rovněž pro nás nejsou důležité osy popisující zeměpisnou polohu, protože nás zde zajímá něco zcela jiného.

1.2.7 Interakce

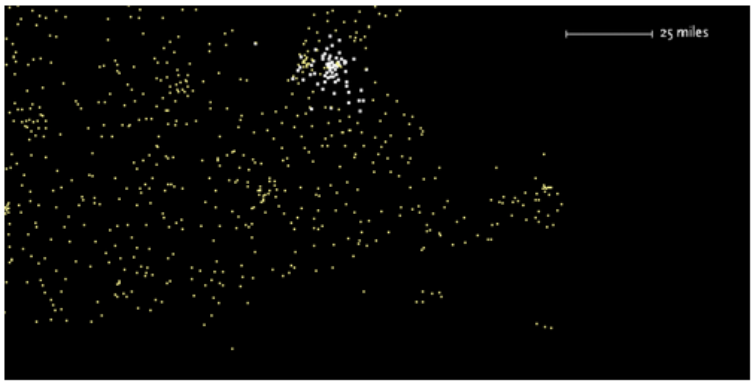
Pokud uživateli chceme nabídnout interaktivní rozhraní je důležité rozmyslet si jej tak, aby bylo pro uživatele intuitivní a v každém případě nematoucí. Rovněž by nemělo nabízet zbytečné množství možností aby neodvádělo od hlavního přínosu vytvořené aplikace. Navíc v příliš komplikovaném rozhraní se uživatelé neorientují a nevyužívají všech jeho možností – stačí jim znát pár základních funkcí.

V tomto příkladu je uživateli poskytnuta možnost zadávat čísla hledaného ZIP kódu, na jehož základě se bíle obarvují oblasti s vyhovujícím kódem. ZIP kód má 5 číslic. S přibývajícím počtem zadaných číslic se upřesňuje (zmenšuje) obarvená oblast. Rovněž je nabídnuta možnost zvětšení (zoom) jednotlivých částí kliknutím na text zoom v pravé dolní části obrazovky.

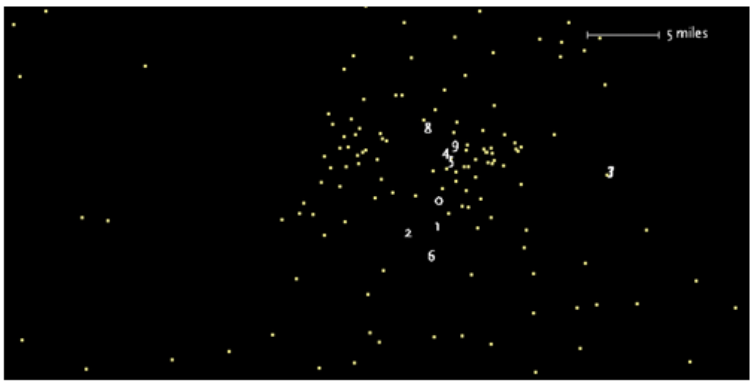
Jelikož bez přečtení návodu na ovládání dané aplikace nemusí být uživateli zřejmé, že může něco psát a tím ovládat chování aplikace, je například vhodné vypsat zprávu, která jej bude informovat o tom, že může zadávat číslice hledaného ZIP kódu.

Pro ukázku chování zde bude uvedeno několik snímků reprezentujících postupné zadávání ZIP kódu také s využitím možnosti zvětšení nalezené oblasti.

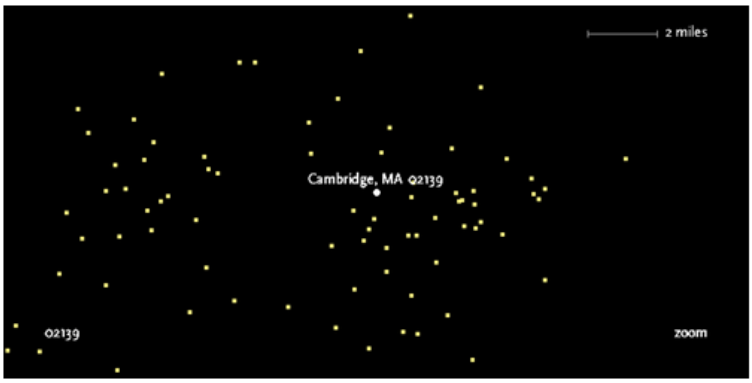
Obrázek 1.5. Výsledná aplikace s výběrem oblasti ZIP 9xxxx [1].



Obrázek 1.6. Výsledná aplikace s využitím možnosti zoom [1].



Obrázek 1.7. Nabídka možných posledních číslic ZIP kódu [1].



Obrázek 1. 8. Zobrazení města podle kompletního ZIP kódu [1].

1.3 Shrnutí

Příklad v této kapitole by měl předvést použití všech fází celkového návrhu v praxi. Tyto fáze nejsou lineárně propojeny, ale podle konkrétního problému mohou tvořit různě propojený graf – některé fáze je třeba opakovat vícekrát, případně může být někdy nutné začít zcela od začátku s jinou množinou dat či lépe navrženou strukturou.

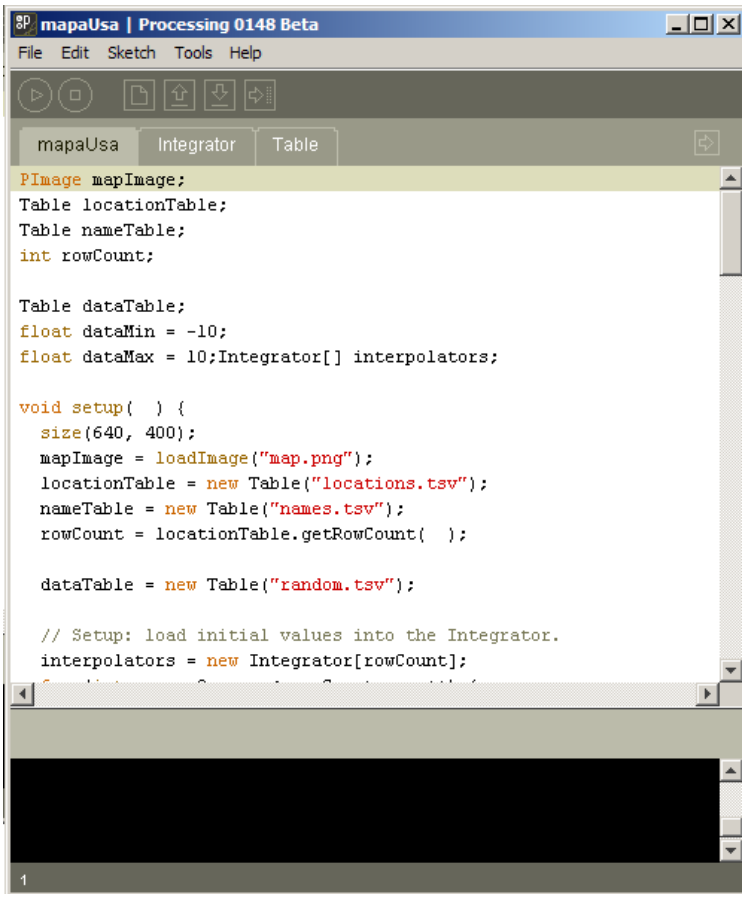
Celkově je tato kapitola základním úvodem do oblasti vizualizace dat a měla by předvést alespoň jeden z možných přístupů k řešení úloh z této oblasti. Navržené fáze a ukázka jejich použití na praktickém příkladu by měla čtenáře nasměrovat na možná odlišný styl přemýšlení než na jaký byl doposud zvyklý.

Nikde není řečeno, že toto je jediná správná cesta. Po přečtení této knihy si myslím, že dodržením těchto vcelku jednoduchých principů vede tato cesta po menším či větším počtu kroků vždy k cíli.

2 Představení aplikace Processing

V této kapitole uvedu stručně základní informace a princip programování v prostředí Processing. Jelikož k této aplikaci existuje velké množství příkladů a taktéž udržovaná dokumentace (www.processing.org), nemyslím si, že je nezbytné ukazovat všechny zdrojové kódy použité v knize. Uvedu jen některé z těch nejzákladnějších, díky nimž by se měl tento nástroj ukázat jako opravdu vhodný i pro programátory - začátečníky.

Processing je jednoduché programovací prostředí navržené k usnadnění vývoje vizuálně orientovaných aplikací, zejména k tvorbě nejrůznějších animací. Je založenu na programovacím jazyku Java a je tak možné jej snadno zahrnout do vlastních aplikací.



Obrázek 2.1. Ukázka prostředí Processing [1].

Processing pracuje vždy s jedním aktivním *skicákem (sketchbook)*, což je kolekce jednotlivých skic či výkresů (*sketch*). V praxi jednu skicu tvoří záložka v editoru, v níž je definována samostatná třída, kterou využívá hlavní program. Na obrázku 2.1. je v aktivní záložce nazvané *mapaUsa* zapsán

hlavní zdrojový kód aplikace, záložky *Integrator* a *Table* jsou využívané třídy pro snadné interpolování hodnot a načítaní tabulkových hodnot.

Jádrem vytvářené aplikace je metoda *setup()*, která má na starosti počáteční inicializaci kreslící plochy a případné nastavení používaných prostředí, a metoda *draw()*, která je cyklicky volána několikrát za sekundu (podle nastavené rychlosti) a je využívána pro vykreslování hodnot. Velmi rychlé cyklické volání velmi snadno umožňuje vytvářet dojem plynulé animace bez jakýchkoliv hlubších znalostí programovacích technik.

Jak je patrné z obrázku 2.1., prostředí Processing obsahuje vlastní jednoduchý editor podporující zvýraznění syntaxe i dodržování formátování bloků. Je možné prostředí nakonfigurovat pro použití s vlastním oblíbeným editorem.

Kromě vlastního spuštění napsaného programu umožňuje toto prostředí snadný export vytvořené aplikace jak do spustitelného souboru či přímo do HTML stránky se zanořeným Java apletem. Pro aplety je ale nutné uvědomit si bezpečnostní omezení jako je například nemožnost přístupu na uživatelský disk, proto musí být použité soubory dostupné ze stejného serveru, na kterém aplet bude umístěn.

Navíc je možné nahrávat jednotlivé snímky běžící aplikace do sekvence obrázků různých formátů a také vykreslovat vektorové obrázky přímo do PDF dokumentu. Existuje spousta zajímavých knihoven, které rozšiřují funkcionalitu o např zpracování xml, propojení s videokamerou a další.

Pro snadnou přenositelnost se zachováním úplného výsledného vzhledu, disponuje toto prostředí také jednoduchým nástrojem pro vytváření fontu, který lze k aplikaci *přibalit* a může být použit i v prostředí, kde požadovaný font ve standardní sadě fontů není k dispozici.

Koncepce tohoto prostředí je dobře promyšlená a je vytvořena elegantně a jednoduše. Celý skicák je uložen ve stejnojmenné složce, v níž se nachází všechny soubory s definovanými třídami (skicy) a složka *data*, která obsahuje veškerá přídavná data jako použité fonty, zdrojová data pro vykreslování apod. Tím je zaručena snadná přenositelnost i bez přímého exportu do jiného formátu.

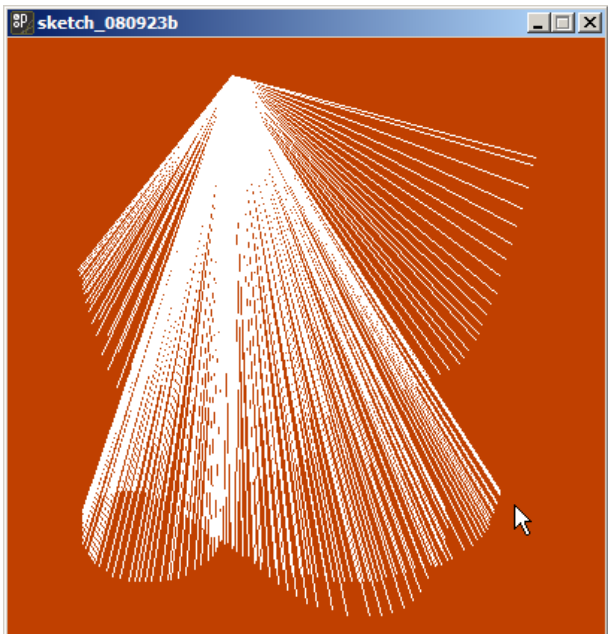
Samozřejmostí je udržovaná on-line dokumentace, která je přímo propojena s vývojovým prostředím, kdy po označení názvu funkce je pod pravým tlačítkem myši dostupná volba nalezení popisu této funkce v referenci.

Pro vývoj rozsáhlejších projektů či využití funkcí tohoto prostředí autor sám doporučuje použití spíše Java IDE. Stačí importovat knihovnu *lib/core.jar* do projektu a vytvořit hlavní třídu, která rozšíří třídu *PApplet*, která zpřístupní všechny funkce prostředí Processing. Dále je nutno dodržovat syntaxi jazyka Java, která již nedovoluje zjednodušený zápis jaký bude uveden v několika ukázkách.

Ukázka zdrojového kódu:

```
void setup() {
  size(400, 400);
  stroke(255);
}

void draw() {
  line (150, 25, mouseX, mouseY);
}

void mousePressed() {
  background(192, 64, 0);
}
```

Tento jednoduchý příklad v prostředí Processing po spuštění vykreslí okno, v němž se vykreslují bílé čáry (volba barvy pomocí metody stroke()) z pozice na souřadnicích (150, 25) do aktuální pozice kurzoru myši. Při stisknutí tlačítka se kreslící plocha přemaže předvolenou barvou.



Obrázek 2.3. Ukázka snímku vytvořené aplikace.

3 Mapa USA

V této kapitole bude naposled předvedena jednoduchost a elegance prostředí Processing pro zobrazení států USA. V dalších částech již nebudou uvedeny žádné úkázky zdrojových textů. Zájemci je mohou nalézt v samotné publikaci nebo najít mnoho ukázek jak na domovské stránce projektu, tak jako přílohy přímo k dodané aplikaci.

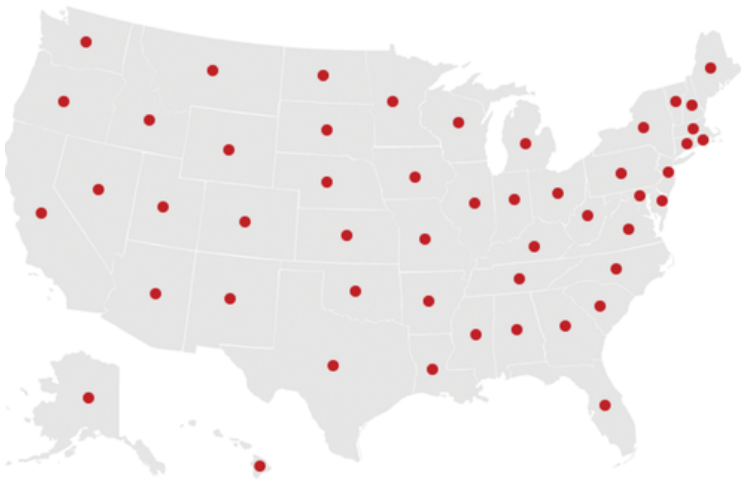
Související soubory:

- http://benfry.com/writing/map/locations.tsv - datový soubor, souřadnice států.
- http://benfry.com/writing/map/Table.pde – třída pro načítání tabulkových dat (knihovna dodávaná autorem, netřeba znát detaily je důležité ji umět použít)
- http://benfry.com/writing/map/map.png – mapa USA jako obrázek na pozadí

Zdrojový kód:

```
PImage mapImage;
Table locationTable; // nutno připojit třídu Table.pde
int rowCount;

void setup(  ) {
  size(640, 400);
  mapImage = loadImage("map.png");
  // vytvoření datové tabulky ze souboru obsahující souřadnice států
  locationTable = new Table("locations.tsv");
  // počet řádků potřebujeme znát dále, proto je globální proměnná
  rowCount = locationTable.getRowCount(  );
}

void draw(  ) {
  background(255);
  image(mapImage, 0, 0);

  // nastavení atributů pro kreslení elips (kruhů)
  smooth(  );
  fill(192, 0, 0);
  noStroke(  );

  // cyklický průchod daty a vykreslování souřadnic pomocí kružnic
```

```
  for (int row = 0; row < rowCount; row++) {
    float x = locationTable.getFloat(row, 1);  // první sloupec dat
    float y = locationTable.getFloat(row, 2);  // druhý sloupec dat
    ellipse(x, y, 9, 9);
  }
}
```



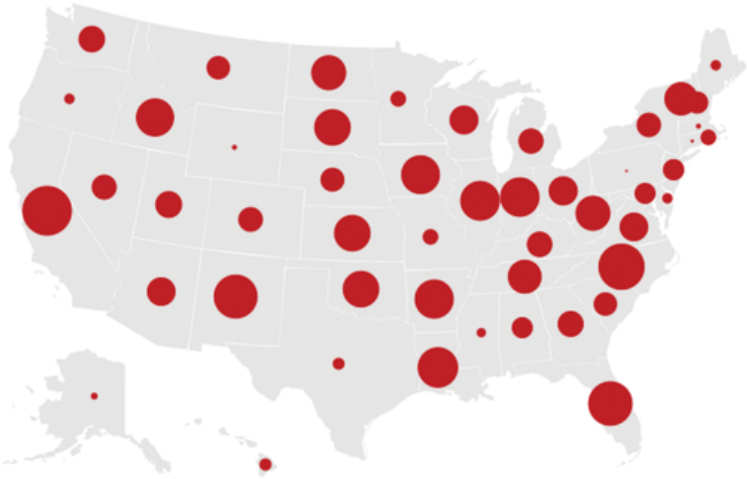
Obrázek 3.1. Výsledek předchozího programu. [1]

Pomocí několika málo řádků zdrojového kódu jsme schopni zobrazit středy států, jejichž souřadnice jsou uloženy v textovém souboru.

Nyní si ukážeme jakým způsobem je možné zobrazit různé hodnoty pro daný stát. Předpokládejme, že máme soubor dat, popisující jistou vlastnost. Například relativní přírůstek/pokles obyvatel za měsíc. Soubor tedy bude obsahovat kladná a záporná čísla.

První grafickou reprezentací může být obrázek 3.2. Vidíme, že rozdílné hodnoty mapujeme na rozdílnou velikost zobrazených kruhů. Důležité pro tento typ zobrazení je **zjistit minimum a maximum všech hodnot a zajistit vhodné mapování** reálné hodnoty na poloměr či průměr zobrazeného kruhu. Jelikož mapování je při zobrazování dat opravdu velmi často používáno, poskytuje Processing řadu vhodných funkcí, které lze využít:

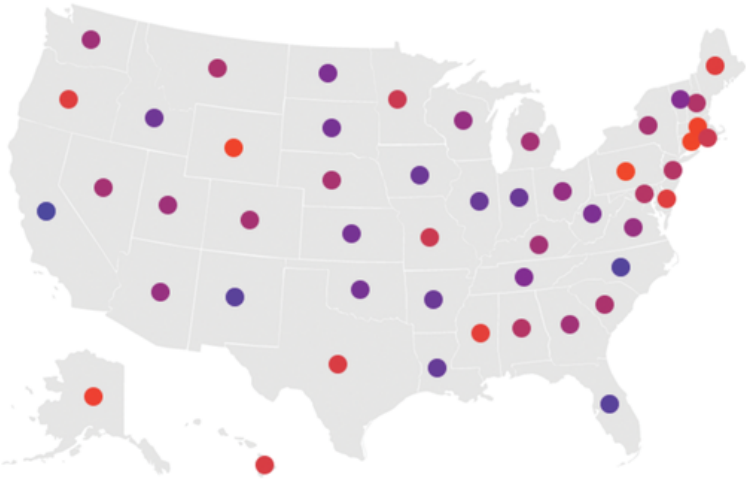
- *map(hodnota, minimum, maximum, vysledne_minimum, vysledne_maximum)* převede *hodnotu* podle poměru mezi původními maximálními rozměry na do nového (výsledného) intervalu hodnot,
- *norm(hodnota, minimum, maximum)* – vrací procentuální vyjádření hodnoty uvnitř zadaného intervalu,
- *lerp(hodnota1, hodnota2, procento)* – mapuje inkrementálně procentuální hodnotu mezi zvolené dvě hodnoty; *leprColor()* je použita pro určení odstínu přechodu mezi barvami dále.



Obrázek 3.2. Zobrazení rozdílných hodnot různou velikostí [1].

Pokud prozatím ponecháme stranou to, že neznáme přesné hodnoty, je tento typ případ toto zobrazení jednou z nejlepších voleb. Data vyjadřují jednu veličinu, proto jsou všechny místa stejně barevná, a rozdíly hodnot jsou vizuálně jasně reprezentovány rozdílnou velikostí (největší poloměr mají největší (kladné) hodnoty, nejmenší zase nejmenší (záporné) hodnoty). Mapování bylo zvoleno na poloměry 2-40 pixelů.

Další z možností je zachovat velikost všech míst a rozdílné hodnoty prezentovat rozdílnými barvami. Pro první ukázku byly jako hraniční barvy zvoleny červená a modrá.

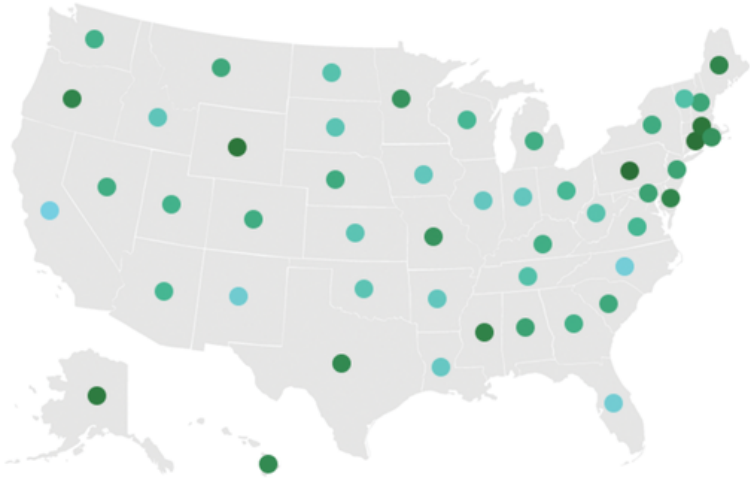


Obrázek 3.3. Mapování hodnot na rozdílné barvy [1].

Oproti prvnímu přístupu vypadá tento výsledek například na můj vkus lépe (působí na mě hezčím dojmem) – možná proto, že si libuji v pravidelnosti. Nicméně vypovídací informační hodnota již není

tak dobrá jako u předchozího což je způsobeno **nevhodnou volbou barev**, protože střední hodnota mezi těmito barvami je fialová – těžko se rozpoznává zda má blíže k modré nebo k červené.

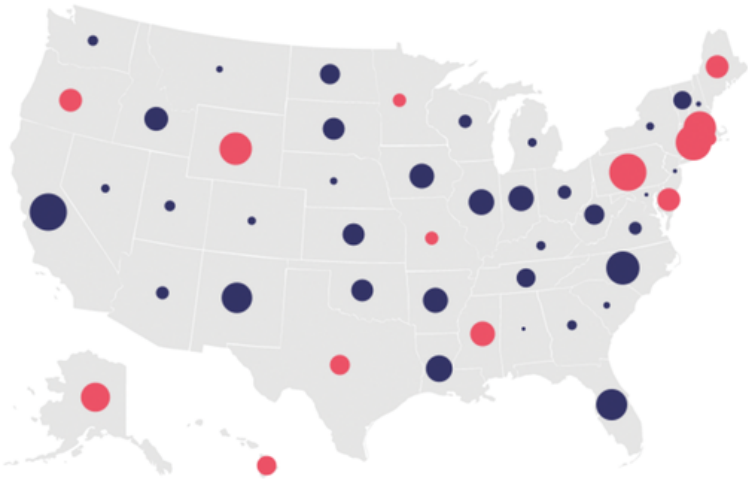
Mnohem lepším řešením je v této situaci použít podobných barev, které nebudou mít zcela odlišnou mezilehlou barvu, nejlépe dvou různých odstínů jedné barvy jako na obrázku 3.4.



Obrázek 3.4. Mapování pomocí dvou příbuzných barev [1].

Zde snáze rozpoznáme rozdílnost barev na základě světlejšího či tmavšího odstínu. Pokud bychom měli k dispozici legendu vysvětlující, že světlejší barva znamená větší přírůstek oproti tmavé barvě znamenající větší deficit, snadno si uděláme představu o změnách v jednotlivých státech.

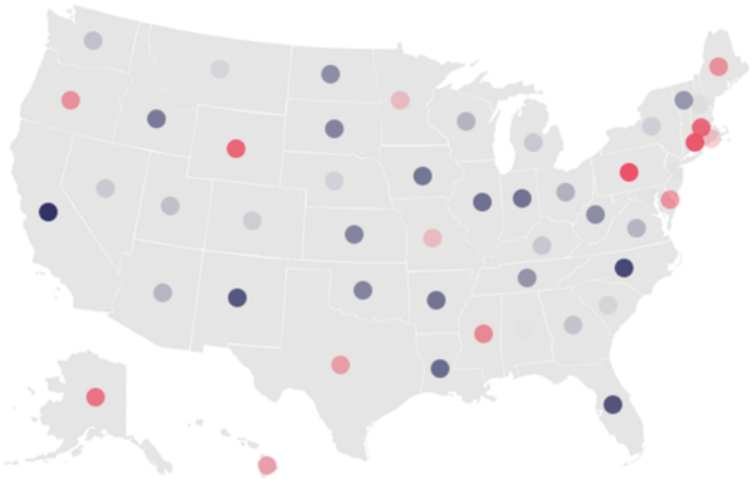
Pro protikladné hodnoty jako v tomto případě (kladná i záporná čísla) je vhodnější volit dvě barvy, kdy jedna vyjadřuje pozitivní hodnoty a druhá negativní. Vlastní hodnotu pak zdůrazníme různou velikostí zobrazených kruhů. V podstatě spojíme první dva přístupy do jednoho a dostaneme tak další z nejvhodnějších reprezentací pro tuto sadu dat.



Obrázek 3.5. Mapování pomocí dvou barev a změny velikosti [1].

Jedinou nevýhodou tohoto zobrazení, které nám v tomto případě vůbec nevadí, je, že zbytečně ztrácíme možnost použít například rozdílné velikosti pro zobrazení jiných veličin. Pokud by barva zobrazovala různé přírůstky obyvatel, pak velikost by mohla reprezentovat objem prodeje alkoholu v daném státě.

Proto do hry může vstoupit ještě průhlednost, kdy zachováme vhodný výběr pouze dvou barev, ale změnu hodnot reprezentujeme různým stupněm průhlednosti barvy. Vizuální dojem je ihned patrný, ale průhledné hodnoty mohou působit *nedůležitým dojmem*. Proto tento typ zobrazení není pro tento příklad také příliš vhodný.



Obrázek 3.6. Mapování hodnot pomocí různých barev a transparence [1].

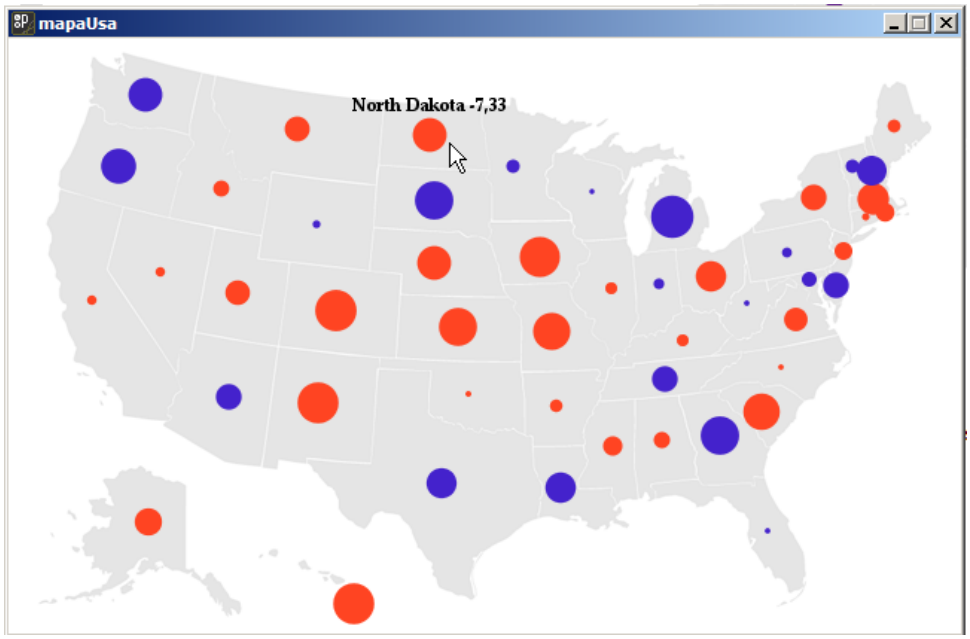
3.1 Zobrazení dalších informací

Pokud již máme zvolen druh základní vizualizace dat, nastal nejvyšší čas dodat vytvářené aplikaci potřebný *šmrnc* a poskytnou uživateli všechny hodnoty, které jsme tak poctivě připravovali (získání, parsování a filtrace hodnot). Opět je nejvhodnějším způsobem interakce s uživatelem, neboť zobrazení textových hodnot je v takovémto zobrazení nesmyslné.

Základní možností interakce je zobrazení názvu či zkratky státu a aktuální hodnoty přírůstku obyvatel po najetí kurzoru myší na zobrazený kruh viz obr. 3.7.

Při použití postupně se měnících hodnot stejného charakteru je také velmi vhodným a efektivním způsobem zobrazit vizuální přechod mezi změnami hodnot. Pokud uživatel může sledovat postupnou změnu (vývoj) hodnot, dokáže si udělat mnohem snáze lepší představu o vývoji dat než při sledování několika po sobě jdoucích, leč oddělených snímků. V ukázkách prostředí Processing se přechody mezi hodnotami řeší pomocí třídy *Integrator.pde*, která je rovněž k dispozici. Tato třída umožní snadnou interpolaci měnících se hodnot a jejich postupné překreslení. V případě změny

vývoje přírůstků obyvatel je v aplikaci interpolace mapována na stisknutí klávesy mezerník, kdy dojde k překreslení v tomto duchu (nelze zachytit do obrázku): Pokud nastal úbytek oproti stávající hodnotě, poloměr kruhu se pomalu zmenšuje až na požadovanou velikost, případně může dojít i ke změně barvy z červené na modrou a naopak, při zvýšeném přírůstku se poloměry zvětšují.



Obrázek 3.7. Interakce s uživatelem.

3.2 Souhrn

Nalezení vhodné reprezentace dat není vždy jednoduchou a přímočarou záležitostí. Pokud ještě nemáme dostatečnou praxi se zobrazením podobných dat je mnohdy je nutné vyzkoušet různé možnosti a hledat správnou cestu metodou pokus omyl.

Důležitých je v tomto rozhodování několik aspektů:

- vytvořit přehlednou reprezentaci dat,
- vytvořit jasnou a pochopitelnou reprezentaci dat,
- vytvořit co nejméně matoucí reprezentaci dat,
- neodvádět uživatelovu pozornost nedůležitými elementy.

Na uvedeném příkladu je patrné, že výsledný dojem může ovlivnit i taková *maličkost* jako volba barvy pro zobrazení hodnot. Ne vždy je správné řídit se vlastním vkusem, ale je nutno vybrat přiměřený počet snadno rozlišitelných barev (také s ohledem na lidi s poruchou barvocitu). Podle uvážení a podle charakteru dat, pokud to přispěje k předchozím třem bodům v seznamu, použít další vizuální prostředky jako různé tvary či velikosti zobrazených bodů, transparence apod.

Z uvedených příkladů je nejlepším řešením obr. 3.2. a obr. 3.5.

4 Za všechno může čas …

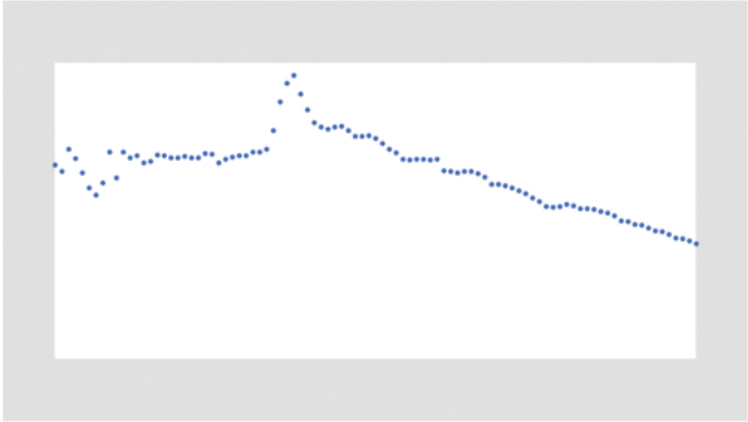
Jelikož se vše s postupem času mění, zobrazení časové závislosti dat je asi nejčastější aplikací vizualizace dat. V této kapitole bude předvedeno několik málo možností prostředí Processing, jak lze zobrazit časové průběhy veličin. Výsledky budou dokonce srovnatelné s kancelářskými balíky MS Excel či Open Office Calc.

Pro vytvoření výsledku je nutno projít znovu několik fází, z nichž většinu v tomto popisu vynecháme a zaměříme se opět spíše na *vylepšování* výsledné reprezentace a zachycení hlavních myšlenek, proč ke změnám dochází. Fáze mohou být následující:

- získání dat (zde opět použita tabulka z textového souboru)
- parsování souboru do použitelné datové struktury
- zjištění/výpočet hraničních hodnot
- nalezení vhodné reprezentace dat a případných jiných možností
- vylepšení reprezentace změnou rozložení, typu, barev, …
- poskytnutí interakce a vizuální interpolace hodnot

V tomto příkladu je zkoumána spotřeba mléka, čaje a kávy v USA různých letech.

S pomocí Processing můžeme snadno zobrazit data získaná ze souboru.

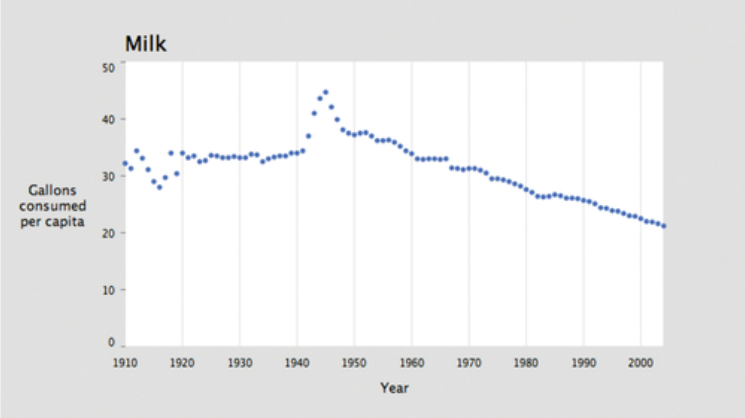


Obrázek 4.1. Prvotní zobrazení dat [1].

Stejně jako nic neříkající obrázek následuje záměrně nic neříkající popis. Tato vizualizace je nám v podstatě k ničemu, pokud neslouží pouze pro naše soukromé účely, kdy víme, co je na ní znázorněno. Pokud výsledek hodláme zpřístupnit širší veřejnosti, nejméně dva zásadní nedostatky:

- chybí pojmenování grafu,
- chybí popisy os pro pochopení významu hodnot.

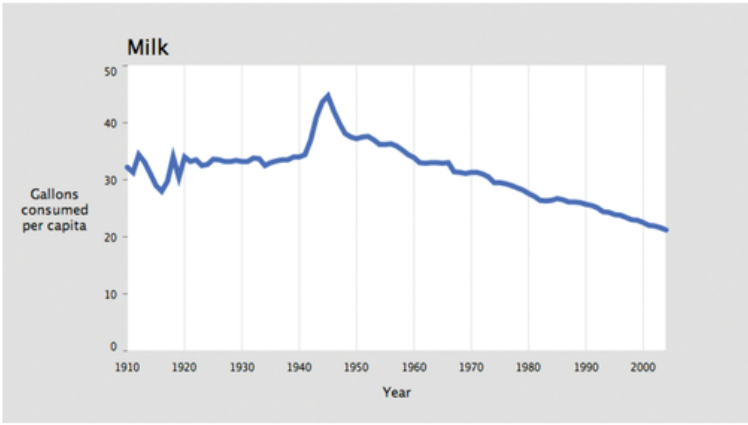
Po odstranění těchto hlavních chyb může graf vypadat následovně.



Obrázek 4.2. Graf spotřeby mléka v jednotlivých letech [1].

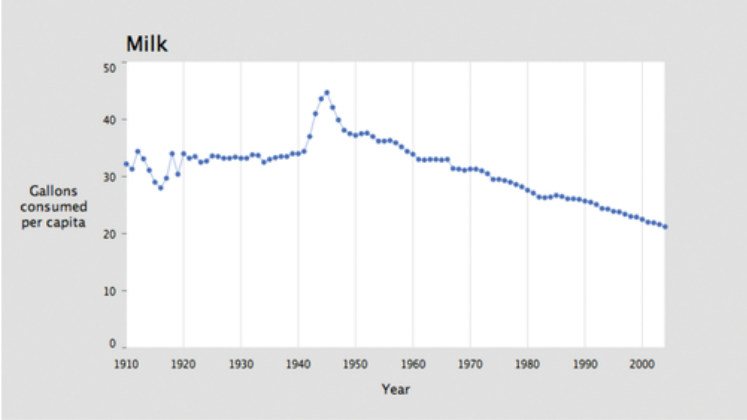
Jelikož podobné grafy lze snadno sestrojit ve známých tabulkových procesorech, není snad nutný žádný rozsáhlejší komentář. Pouze zmínka k popisu horizontální osy – **je dobré popisovat osu vodorovně, což je přirozené pro čtení.** Rozdělení delšího textu do několika řádků je pro koncové uživatele přívětivější než rotace textu, která může znesnadnit čtení pokud se nejedná o vytištěné materiály, které lze snadno natočit.

Pro spojité hodnoty se používá spojitý graf – Processing disponuje pro tvorbu křivek jak definicí postupných vrcholů, tak zadáním tvarů křivky.



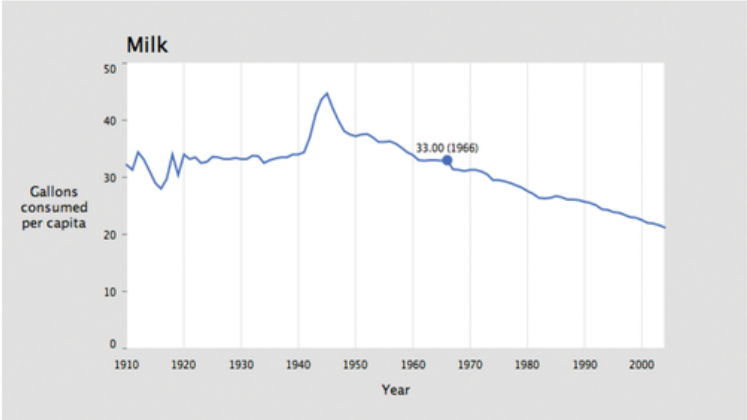
Obrázek 4.3. Zachycení spojité podstaty dat [1].

Pomocí různé tloušťky čáry můžeme upravit výsledný vzhled grafu tak, aby byly viditelné spolehlivé hodnoty (které opravdu známe) i průchod aproximační křivky.



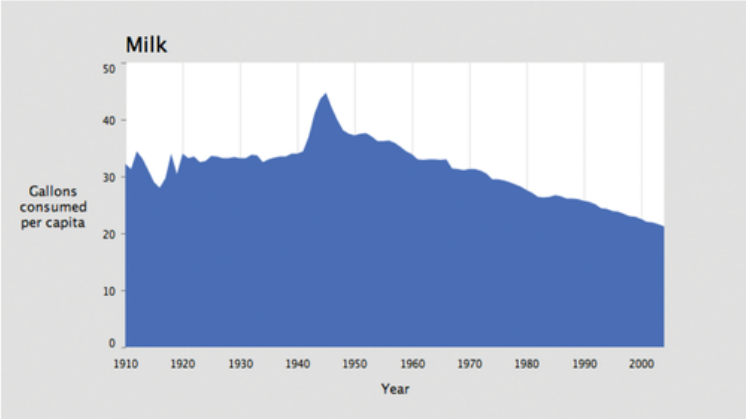
Obrázek 4.4. Spojitý graf s menší tloušťkou čáry [1].

V rámci uživatelské přívětivosti je velmi vhodné zkombinovat krásu hladké křivky grafu s možností interakce pro přesné odečtení zobrazených bodů. Při najetí kurzorem myši k oblasti křivky se zvýrazní nejbližší přesný bod (uložený v datech) s popisem zobrazujícím konkrétní hodnoty (zachyceno na obr. 4.5.).



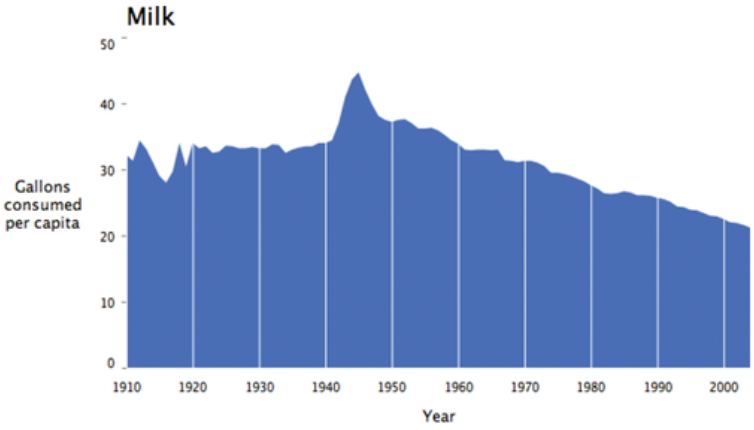
Obrázek 4.5. Zapojení interakce do spojitého grafu [1].

Atraktivnější a v některých případech vhodnější jsou plošné grafy. Jsou vhodné pokud zobrazujeme určitý *objem* sledované veličiny, kdy je taktéž stabilní spodní hodnota – jakožto například zde spotřeba mléka či graf srážek. Není vhodné například pro zobrazení grafu teplot. Výsledný graf je vidět na obrázku 4.6.



Obrázek 4.6. Plošný graf [1].

Jelikož plocha grafu působí dojmem pozadí, je vhodné odstranit zbytečné rušivé elementy. Upravená verze grafu je na obrázku 4.7. Dále je možné různě experimentovat a hledat nejvhodnější vizualizaci. Samozřejmostí je zapojení interakce, která nám snadno pomůže vyřešit původní problém – zobrazení spotřeby mléka, čaje a kávy. Prozatím jsme zobrazovali pouze spotřebu mléka. Sloučit všechny tři grafy do jednoho lze například pomocí vytvoření záložek pro přepínání mezi jednotlivými grafy s přidáním interpolace hodnot, kdy si uživatel snadno všimne vzrůstu či poklesu spotřeby při přepnutí na graf dané suroviny.



Obrázek 4.7. Vylepšený plošný graf bez rušivých elementů [1].

5 Různé příklady zobrazení dat

V této části textu předvedeme některá další zajímavá řešení příkladů z knihy. Postup tvorby se v podstatě neustále opakuje – z připravených dat je vytvořena jejich prvotní vizualizace, která je nadále v jednotlivých krocích vylepšována, až do co nejlepší možné podoby.

Zajímavé řešení vymyslel autor knihy pro **zobrazení výsledků baseballové ligy**, kdy chtěl rovněž ukázat rozpočty jednotlivých týmů takovou formou, aby bylo názorně předvedeno, zda-li se *investice* do týmu vyplatila či nikoliv. Získané hodnoty tvoří vlastně uspořádané seznamy. Otázkou zbývá jakým způsobem tyto seznamy vhodně reprezentovat?

Původní návrh je vidět na obrázku 5.1. Důležitou součástí seznamu týmů a jejich výsledků jsou také loga týmů – zejména z důvodu, že se podle nich fanoušci baseballu orientují mnohem snáze a rychleji než při čtení samotných názvů. Hodnoty jsou vykresleny ve dvou sloupcích vždy sestupně – týmy podle poměru výher a proher a rozpočty podle počtu utracených dolarů.

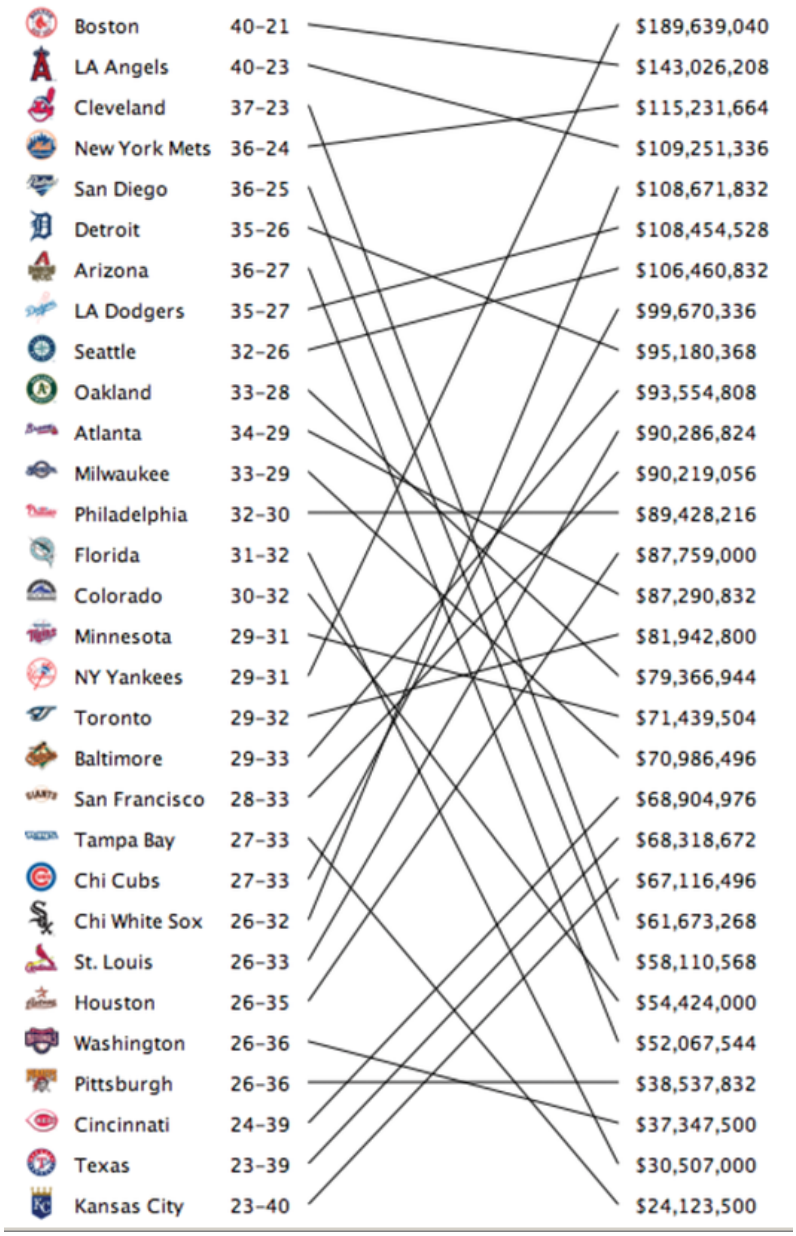
Provázání těchto dvou oddělených seznamů je řešeno spojením pomocí linek, kdy je možné vysledovat, který tým z dané investice nejvíce vytěžil či kde se jednalo ryze o ztrátu peněz. První dva týmy evidentně svůj rozpočet využili správně, kdežto tým NY Yankees s největším rozpočtem skončil ve výsledku až někde v polovině!

Nevýhodou této reprezentace je stížená orientace ve směsici křížících se čar (např. pro týmy Chi White Socks, či Chi Cubs). Je třeba řešit otázku vylepšení zvolené vizualizace. Jako první možností, která člověku přijde na mysl je barevně rozlišit čáry – zbývá si rozmyslet kolik barev použít, jakým systémem zajistit jejich střídání apod. Obecným pravidlem je, **že by se nemělo při vizualizaci používat příliš mnoho barev. Nejvýše pět či šest.** Pokud potřebujeme více barev je lepší použít různé barevné odstíny vybraných základních barev.

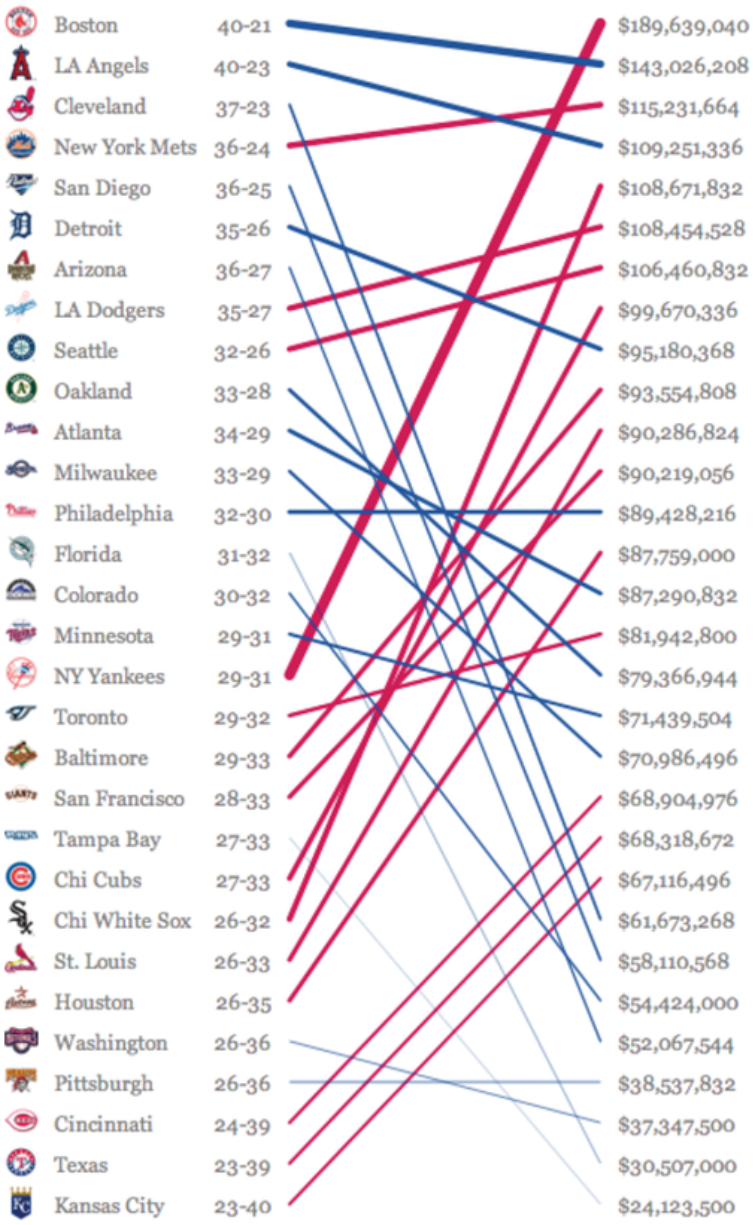
Pro náš případ zcela postačí barvy dvě – jedna barva bude pro čáry sestupné tendence (kdy týmy jakoby předčili očekávání a umístili se výše než je znázorněna výše rozpočtu) a druhá barva pro čáry opačného směru.

Toto řešení celý graf zpřehlední, nicméně pro další zvýraznění hlavní myšlenky prezentovaných dat lze použít různé tloušťky čar, které jsou přímo úměrné výši rozpočtu každého týmu.

I přes výrazné zlepšení a usnadnění orientace působí texty stále trochu rušivě, proto autor zvolil jejich zabarvení do šedé a také použil sympatičtější typ písma. Zájemce o výsledky bude stále snadno navigován pomocí loga týmu o který se zajímá a pomocí spojnice snadno přejde na rozpočet daného týmu. Výsledná prezentace je zobrazena na obrázku 4.2.

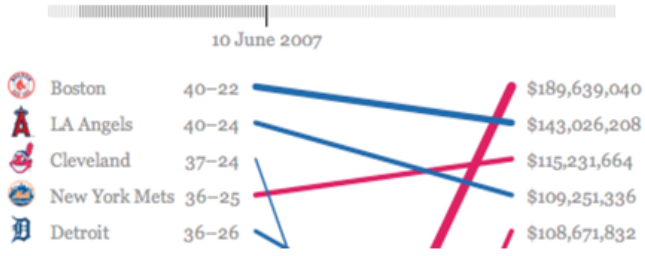


Obrázek 4.1. Původní návrh prezentace výsledků basebalové sezony [1].



Obrázek 4.2. Výsledná reprezentace [1].

Tímto samozřejmě možnosti vylepšování nekončí. Tato data jsou opět vhodná pro využití interakce s ohledem na jejich časovou závislost. **Při práci s časovými daty je nutné se vždy vypořádat s možnými problémy, které mohou nastat jako jsou zejména chybějící či nesouvislá data.** Forma interakce v tomto příkladě je zvolena jako časový posuvník průběhem sezony, kdy se postupně tvoří výsledky jednotlivých her a mění se pořadí týmů (naznačeno obrázkem 4.3.).



Dalším zajímavým typem druhem zobrazení hierarchických dat jsou uspořádané mapy – **treemaps**. Princip *treemap* je zobrazení stromu pomocí čtvercových (či obdélníkových oblastí), kdy každá větev se dělí na další čtverce, pokud obsahuje další podvětve. Tuto koncepci předložil Ben Shneiderman na počátku devadesátých let.

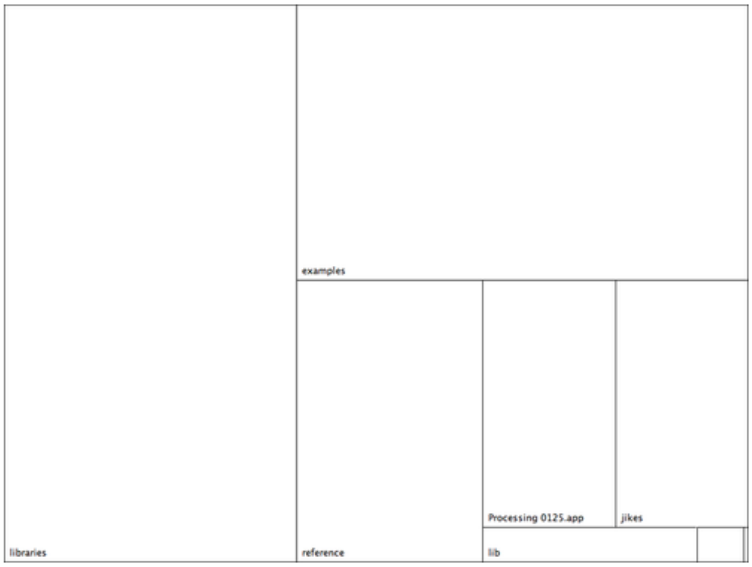
Nejlépe bude předvést toto zobrazení na konkrétním příkladě, kdy je použita pro zobrazení relativní frekvence slov použitých v knize Marka Twaina.

Obrázek 4.4. Ukázka treemap [1].

Pro takovou množinu dat, není možná volba zobrazení právě nejšťastnější, protože kromě hodně frekventovaných slov jako *this, and, of, ...* jsou vykresleny pouze neidentifikovatelné čtverečky. Samozřejmě pomocí interakce uživatele, kdy by se po najetí kurzoru myší nad zkoumaný čtvereček zobrazilo slovo, které reprezentuje a jeho frekvence, byla by tato reprezentace snad dostačující. Při takto drobném rozčlenění by byl ale problém už se samotným zaměřením těch nejmenších částí.

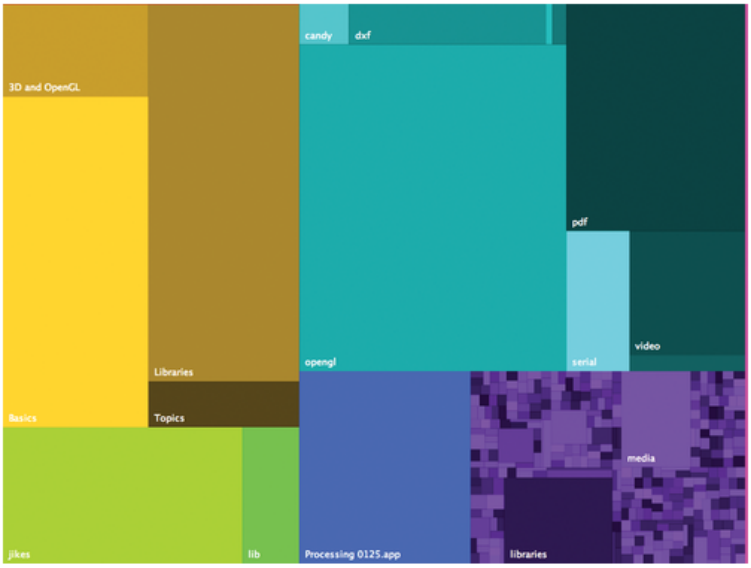
Myšlenka treemap se ale dobře uchytila a je používána pro reprezentaci využitého/volného místa na pevném disku počítače (použito například v aplikaci SpaceMonger, http://www.sixty-five.cc/sm/).

V prostředí Processing lze pomocí třídy pro práci s treemapami snadno vytvářet toto zobrazení jak v jednoduchém základním tvaru,



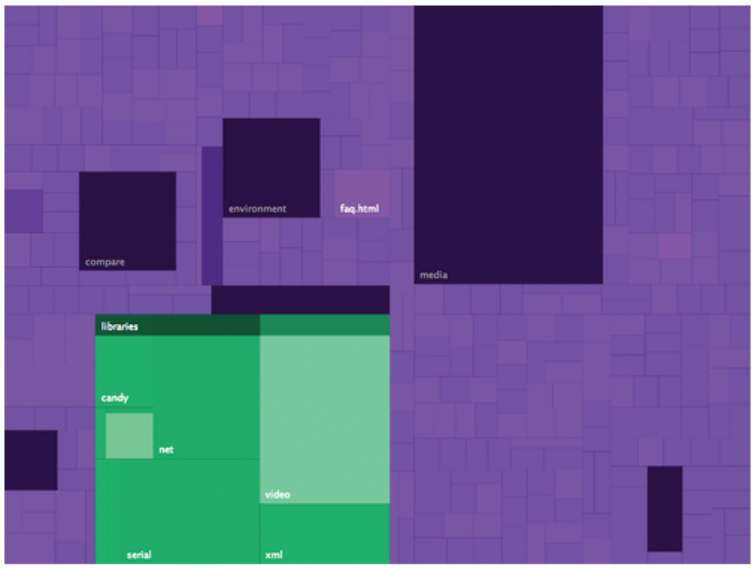
Obrázek 4.5. Rozložení místa na disku formou treemap [1].

či vylepšeného barevného modelu pro lepší přehlednost (znovu připomínám myšlenku nepoužívat příliš mnoho barev, namísto toho je použito tří odstínů od každé použité barvy).



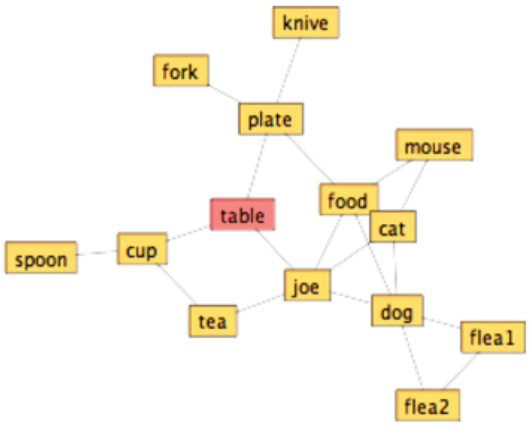
Obrázek 4.6. Barevná treemapa [1].

Do hry lze opět zapojit i interakci a vytvořit zvětšení vybrané oblasti, tak aby bylo možné zkoumat zkutečný obsah všech i hlouběji zanořených složek. U této možnosti je použita pouze jedna základní barva a její různé odstíny a jedna barva pro zvětšenou oblast. Tím na uživatele nepůsobí dojmem různých nezávislých oblastí jako na obrázku 4.6.



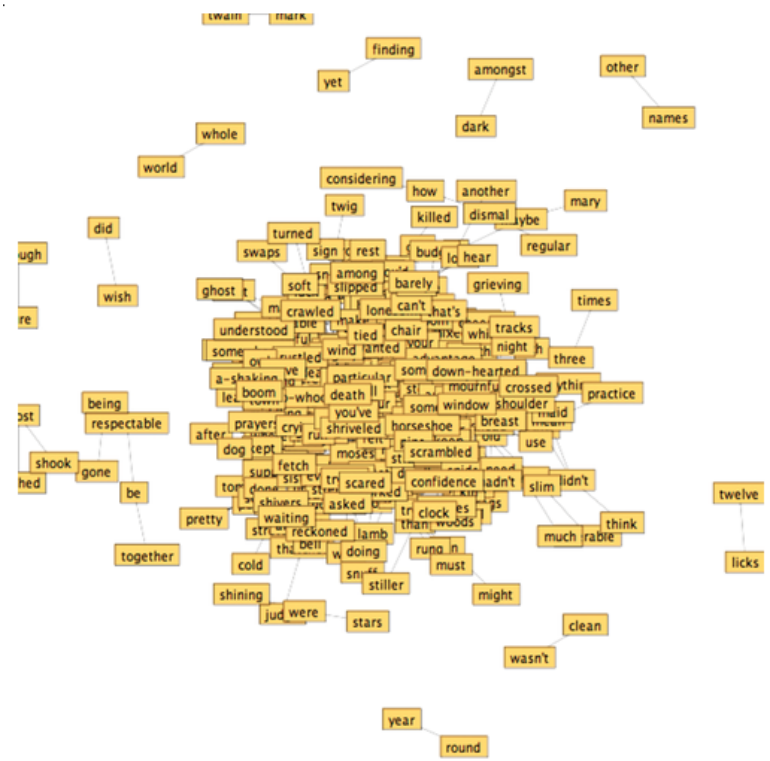
Obrázek 4.7. Využití místa jednotlivých složek s možností *zoom* [1].

Grafy. V prostředí Processing není problém vytvořit jednoduchý graf propojených uzlů.



Obrázek 4.8. Jednoduchý graf [1].

Pokud bychom však tímto způsobem chtěli zobrazit vztahy mezi slovy v textu, kdybychom frekventovanější spojení slov chtěli zobrazit více ve středu grafu, dostaneme nepřehledný a chaotický shluk uzlů, jehož praktické využití je takřka nulové a nepomáhají ani různé snahy o zpřehlednění (obrázek 4.10.). Zobrazených informací je zkrátka příliš mnoho.



Obrázek 4.9. Nevhodné použití grafu pro modelování blízkých slov [1].

Obrázek 4.10. Pokus o zpřehlednění – zrácíme však význam slov [1].

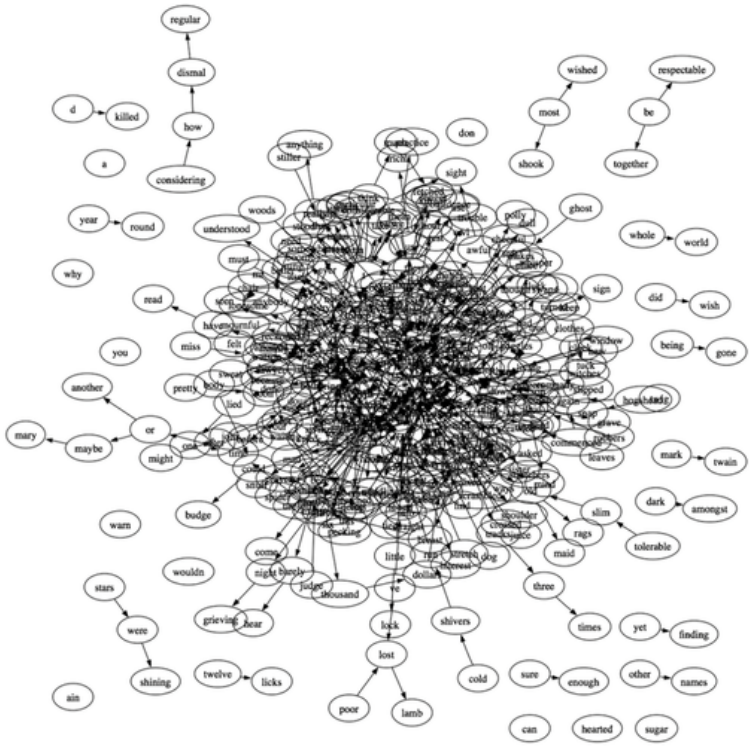
V takovémto případě nastal správný čas pro zamyšlení nad tím, zda-li jsme opravdu zvolili správnou cestu pro prezentaci dat. **Je opravdu vždy nutné prezentovat výsledky formou obrázku? Nebyla by plně dostačující a také přehlednější obyčejná tabulka?**

Při hledání vhodnějšího způsobu pro vykreslení dat je také možné poohlédnout se po jiných projektech zaměřených tímto směrem. Jako velice zajímavý nástroj je v knize zmíněn nástroj **graphviz** (http://www.graphviz.org), který je schopen vykreslit propojení i velmi rozsáhlých sítí. Obrázky 4.11. a 4.12. ukazují možné výsledky, které ovšem nejsou příliš čitelnější než výše uvedené řešení. Graphviz využívá zajímavý interní formát pojmenovaný **dot**, což je jednoduchý textový formát pro definici sítě.

Ukázka formátu dot pro graf z obrázku 4.8.

```
digraph {
    joe -> food;
    joe -> dog;
    joe -> tea;
    joe -> cat;
    joe -> table;
    table -> plate;
    plate -> food;
    food -> mouse;
    food -> dog;
    mouse -> cat;
    table -> cup;
    cup -> tea;
    dog -> cat;
    cup -> spoon;
    plate -> fork;
    dog -> flea1;
    dog -> flea2;
    flea1 -> flea2;
    plate -> knife;
}
```

Obrázek 4.9. Výsledek vykreslený jako síť pomocí *Graphviz* [1].



Obrázek 4.10. Jiný způsob zobrazení pomocí Graphviz [1].

Jako poslední zajímavou myšlenku bych zmínil **silou řízené rozložení (*force-directed layout*)**. Jedná se o metodu pro získání vhodného rozložení uzlů pomocí působením přitažlivých a odpudivých sil. Definováním nových uzlů dochází k rekonfiguraci uspořádání tak, aby bylo dosaženo ideálního tvaru.

6 Závěr

Tento text se snažil vystihnout hlavní myšlenky a principy, které lze získat po přečtení knihy *Visualizing Data* a které by čtenář nadále mohl uplatnit v libovolné úloze vizualizace dat. Není nezbytně nutné používat představený nástroj *Processing*, i když působí velmi dobrým dojmem, ale hlavně je zapotřebí naučit se správně postupovat a přemýšlet již ve fázi návrhu řešení konkrétního problému. Doufám, že k tomu text bude alespoň trochu nápomocen, zejména ukázkami nejrůznějších příkladů a obrázků jejich řešení, kdy je snaha o zachycení postupné evoluce od prvotního návrhu až k finální aplikaci. Při dodržení několika jednoduchých základních pravidel by neměl být problém navrhnout vhodný způsob reprezentace datové množiny i pro čtenáře s nedostatkem grafického jemnocitu.

Jako příloha je uveden seznam užitečných nástrojů jak pro vytváření grafiky tak pro usnadnění prvotních fází získávání a parsování dat, které byly zmíněny v textu knihy.

Reference

[1] Ben Fry *Visualizing Data.* O'Reilly Media, Inc., 2008, ISBN: 978-0-596-51455-6

Příloha

- **Přehled nástrojů a odkazů:**
  - http://processing.org – domovská stránka projektu Processing.
  - http://benfry.com/writing – autorova stránka věnovaná publikaci
  - http://www.jfree.org/jfreechart – knihovna pro tvorbu grafů (Java)
  - http://www.cs.umd.edu/hcil/treemap-history/Treemaps-Java-Algorithms.zip
  - http://www.graphviz.org – nástroj pro kreslení grafů a sítí
  - http://cygwin.com – Unixové prostředí pod systémem Windows
  - http://xmlgraphics.apache.org/batik – parsování SVG souborů
  - http://tidy.sourceforge.net – zvalidnění HTML kódu (pro následné snazší parsování)
  - http://htmlparser.sourceforge.net – HTML parser
  - http://xerces.apache.org/xerces-j – XML parser (Java)