ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

Študijný odbor: **Informatika**

Oľga Chovancová

Vizualizácia dát získaných pomocou SCADA systémov s využitím HTML 5 štandartov

Vedúci: **Ing. Juraj Veverka**

Tútor **Ing. Patrik Hrkút, PhD.**

Reg.č. 5/2014 Máj 2015

Abstrakt

CHOVANCOVÁ OĽGA: *Vizualizácia dát získaných pomocou SCADA systémov s využitím HTML 5 štandartov* [Bakalárska práca]
Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky, Katedra softvérových technológií.
Vedúci: Ing. Juraj Veverka
Stupeň odbornej kvalifikácie: Inžinier v Telecommunication, Electrical Engineering 1995 – 2000 todo Solution Design Architect v Ipesofte TODOO
Tútor Ing. Patrik Hrkút, PhD.

Obsahom práce je vzorová sada grafických komponentov na vizualizáciu technologických procesov s využitím HTML 5 štandardov. Jedná sa o grafické komponenty, ktoré nie sú bežne dostupné na tvorbu interaktívnych webových aplikácii ako napríklad vizualizácie mechanických súčasti hydraulických systémov, technologických liniek, silových a výkonových častí automatizačných sústav. Návrh interface, pomocou, ktorého budú tieto komponenty komunikovať so serverovou časťou SCADA systému. Cieľová platforma pre výslednú webovú aplikáciu bude kompatibilná s rodinou štandardov HTML 5 pre každý webový prehľadávač.

Abstract

CHOVANCOVÁ OĽGA: *Data visualization acquired by SCADA systems using HTML5 standarts*
[Bacalar thesis]
University of Žilina, Faculty of Management Science and Informatics, Department of TODO.
Tutor: Ing. Juraj Veverka.
Qualification level: Engineer in field ..... Žilina: TODO
TODO

The main idea of this ... TODO

Prehlásenie

Prehlasujem, že som túto prácu napísala samostatne a že som uviedola všetky použité pramene a literatúru, z ktorých som čerpala.

V Žiline, dňa DD.MM.2015 Oľga Chovancová

Obsah

obsaa

Zoznam obrázkov

Zoznam tabuliek

Úvod

Téma práce je vizualizácia technologických dát zo SCADA systémov na webe. Cieľ práce je nájsť postup tvorby a vizualizácie grafických komponentov. Produktom bakalárskej práce je grafický komponent na vizualizáciu technologických procesov s využitím HTML 5 štandardov. V práci je animovaná prečerpávacia stanica. Navrhnutý bol aj interface, pomocou ktorého komponenty komunikujú so serverovou časťou SCADA systému.

Návrh interface, pomocou ktorého budú tieto komponenty komunikovať so serverovou časťou SCADA systému.

V súčasnosti je v IPESOFT s.r.o. software, ktorý dokáže vizualizovať dáta z technológii pomocou "hrubých klientov", čo sú natívne (exe) Windows aplikácie a je technológia, ktorá dokáže rovnaké dáta zobrazovať na webe.

Aktuálna webová prezentácia takýchto dát nespĺňa súčasne štandardy pre moderne webové aplikácie a preto je potrebne nájsť nový spôsob vizualizácie na webe, ktorá bude v budúcnosti použiteľná na rôznych platformách, nielen na PC.

Výsledná webová aplikácia je kompatibilná s štandardmi HTML 5. Riešenie využíva výhradne open-source knižnice s licenciami typu MIT, GNU GPL, BSD, Apache 2. Zdrojové kódy práce sú udržiavané v Git repository. `https://github.com/chovancova/project`

Postup práce:

1. Analýza požiadaviek, prieskum možnosti využitia WYSIWYG

   editorov na tvorbu grafických komponent s možnosťou exportu do formátov SVG, JSON, XML, alebo JavaScript.

2. Výber vhodných open-source knižníc na tvorbu grafických komponent kompatibilných

s HTML 5.

3. Návrh REST API na prepojenie grafických komponent so SCADA serverom.
4. Analýza možnosti automatického mapovania API grafických prvkov pomocou metadát na existujúce API dostupné pre SCADA server D2000.
5. Popis postupu vizualizácie grafického komponentu
6. Implementácia vzorovej sady grafických komponent.
7. Analýza výkonnosti a výkonnostné obmedzenia.

Kapitola 1

Základné pojmy

1.1 HTML 5 štandardy

Medzi štandardy ...

Na obrázku 1.1 sú HTML5 rozhrania API a súvisiace technológie taxonómie a status

1.2 Čo je SVG?

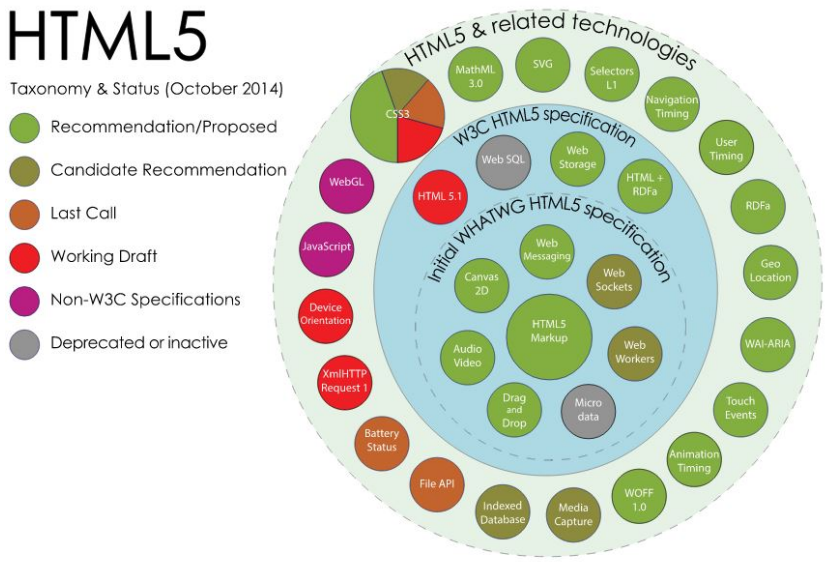
Scalable Vector Graphics (SVG) je štandardný formát pre vektorovú grafiku. Vektorová grafika je definovaná cez body, priamky, mnohouholníky, elipsy, krivky, alebo iné geometrické tvary.

SVG je jazyk na opísanie dvojrozmernej grafiky v EXtensible Markup Language (XML). Vďaka tomu, umožňuje reprezentáciu grafických informácii v kompaktnom a prenositeľnom tvare.

SVG povoľuje tieto tri typy grafických objektov: vektorové grafické tvary, obrázky a text. Grafické objekty môžu byť zoskupené, štylizované, zmenené, a kombinované do predošlých vrstiev objektov.

SVG obrázky môžu byť dynamické a interaktívne. Document Object Model (DOM) pre SVG, ktoré zahŕňa celé XML DOM, a povoľuje priamočiaro a efektívnu vektorovú grafickú animáciu cez príkazy. TODO

Prispôsobiteľnosť SVG umožňuje zmeniť veľkosť grafického komponentu bez straty kvality



Obr. 1.1: HTML 5 API

vzhľadu. Čo umožňuje zobraziť responzívne na viacerých možných zariadení. SVG sa bude zobrazovať rovnako na rôznych platformách. Je kompatibilná s štandardmi HTML5, ktoré navrhla World Wide Web Consortium (W3C).

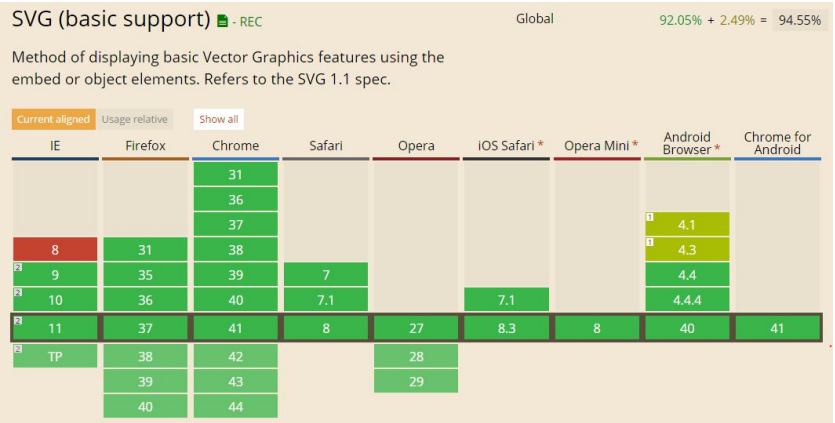
1.2.1 Podpora v prehliadači

Súčasné prehliadače plne podporujú SVG elementy. V tabuľke je TODO - PREROBIM TO ESTE DO TABULKY

1.2.2 Rozdiely medzi SVG a Canvas

TODO SVG je jazyk na opisanie dvojrozmernej grafiky v XML. Canvas kreslí dvojrozmernú grafiku za behu programu cez JavaScript. SVG je XML založený, čo znamená, že každý element je dostupný cez SVG DOM. Môžem k tomu priložiť JavaScript na ovládanie udalostí elementov. TODO . V SVG je každý tvar zapamätaný ako objekt. Ak potrebujem zmeniť atribút SVG, tak prehliadač automaticky prekreslí daný tvar.

Canvas je prekresľovaný pixel za pixelom. Prehliadač na neho zabudne, ako náhle sa vykreslí. Keď chcem zmeniť jeho pozíciu, musím prekresliť úplne všetko.



Obr. 1.2: podpora SVG vo webových prehliadavačoch

1.2.3 Porovnanie Canvas a SVG

Tabuľka 1.1 zobrazuje niekoľko dôležitých odlišností medzi Canvas a SVG. TODO DPI

Canvas	SVG
Rozlíšenie závislé	Rozlíšenie nezávislé
Nepodporuje manipuláciu udalostí	Podporuje manipuláciu udalostí
Malé vykresľovacie možnosti	Vhodné pre aplikácie s veľkými vykresľovacími plochami
Možnosť uložiť výsledok ako .png, .jpg	Pomalé vykresľovanie, ak je to komplexné
Veľmi vhodné pre grafické-intenzívne hry	Nevhodné pre hracie aplikácie

Tabuľka 1.1: Porovnanie Canvas a SVG

1.3 Základná syntax SVG

V HTML5 sa môžu používať vložené SVG elementy priamo v na HTML stránke.

TODO DOPLNIT POUTIZIIE TRI SPOSOBY `https://css-tricks.com/using-svg/`

1.3.1 Príklad použitia SVG v HTML

HTML kód:

```
<!DOCTYPE html>
<html>
<head lang="sk">
<meta charset="UTF-8">
<title></title>
</head>
<body>

  <svg width="100" height="100">
    <circle cx="50" cy="50" r="40" stroke="black" stroke-width
    ="2" fill="silver" />
  </svg>

</body>
</html>
```

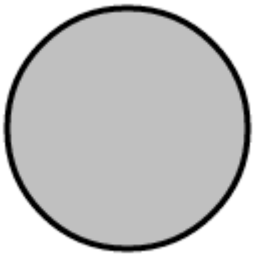
Popis SVG kódu

SVG obrázok začína s <svg> elementom. Atribúty elementu <svg> sú width a height. Definujú šírku a výšku SVG obrázka. Element <circle> je použitý na nakreslenie kruhu. Atribúty cx, cy definujú x, y súradnice od centra kruhu. Ak je cx, cy vynechané, tak center kruhu je nastavený na $(0, 0)$. Atribút r definuje polomer kruhu. Atribúty stroke a stroke-width určujú to ako bude vyzerať obrys útvaru. Kruh má nastavený 2px čierny okraj. Atribút fill vyplní vnútro kruhu. V príklade je vyplnený sivou farbou. Tag, ktorý uzavrie SVG obrázok je </svg>. Keďže SVG je validné XML, tak všetky elementy musia byť správne zatvorené.

Vykreslí na HTML webovú stránku útvar, ktorý je na obrázku 1.3.

1.4 SVG útvary

SVG má preddefinované tieto tvary elementov:



Obr. 1.3: Vykreslenie SVG na HTML stránke

- Obdĺžnik <rect>
- Kruh <circle>
- Elipsa <ellipse>
- Čiara <line>
- Polyline <polyline>
- Mnohouholník <polygon>
- Cesta <path>

TODO ESTE NIECO PRIDAT

Kapitola 2

Analýza požiadaviek

Kapitola popisuje výber z dostupných nástrojov a knižníc.

2.1 Nástroje na tvorbu grafických komponentov

WYSIWYG editory, ktoré umožňujú tvorbu grafických komponentov sú:

- Adobe Illustrator,
- CorelDraw,
- Inkscape,
- Sketch,
- `http://www.drawsvg.org/` .

Nástroj, ktorý najviac vyhovuje požiadavkam je Inkscape. Adobe Illustrator, CorelDraw, Sketch boli platené. TODO PREFORMULOVAT

2.2 JavaScriptové knižnice pre grafické komponenty

Na internete sa nachádzajú tieto OpenSource JavaScriptové knižnice na tvorbu grafických komponentov:

- D3.js,
- Raphael.js,
- Snap.svg.js,
- Svg.js.

Popis jednotlivých JavaScriptových knižníc.

2.2.1 D3.js

D3.js je JavaScriptová knižnica určená na manipuláciu dokumentov vychádzajúcich z dátach. Pomocou HTML, SVG a CSS umožňuje ...TODO TODO TODO Je vhodná na vytváranie interaktívnych SVG grafov s hladkými prechodmi a interakciami.

D3 rieši efektívnu manipuláciu dokumentov zakladajúcich si na dátach. Využíva webové štandardy ako HTML, SVG a CSS3. [12]

2.2.2 Raphaël.js

Raphaël je malá JavaScriptová knižnica, ktorá umožnuje jednoducho pracovať s vektorovou grafikou na webe. Umožňuje pomocou jednoduchých príkazov vytvárať špecifické grafy, obrázky.

Raphaël využíva SVG W3C odporúčania a VML na tvorbu grafických komponentov. Z toho vyplýva, to že každý grafický objekt, ktorý vytvorím je zároveň aj DOM objekt. To umožnuje cez JavaScriptové pridávať manipuláciu udalostí, alebo upravovať ich neskôr. Momentálne podporuje Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.[11] Autor knižnice je Dmitry Baranovskiy. Raphael API má už ukončený vývoj, ale používatelia ho stále používajú. TODO Knižnica neumožnuje load SVG dokumentu zo súboru.

2.2.3 Snap.svg.js

Snap.svg.js je JavaScriptová knižnica na prácu s SVG. Poskytuje pre webových developerov API, ktorá umožňuje animáciu a manipulovanie s buď existujúcim SVG, alebo vygenerovaným s Snapom.

TODO - NASTUPCA RAPHAELA TODO - PRAGRAMTIC SVG CREATING TODO - LOAD EXISTING SVG OBJECT

Snap bol napísaný rovnakým autorom ako Raphael. Bola navrhnutá špeciálne pre moderné prehliadače (IE9 a vyššie, Safari, Chrome, Firefox, and Opera). Z toho vyplýva, že umožňuje podporu maskovania, strihania, vzorov, plných gradientov, skupín...

Medzi hlavnú výhodu je schopnosť pracovať s existujúcim SVG súborom. To znamená, že SVG obsah generovať cez Snap API, aby SA MOHOL ODDELENE používať. Z toho vyplýva, že SA DÁ vytvoriť SVG obsah v nástroji ako Illustrator, Inkscape, alebo Sketch a potom animovať, alebo inak manipulovať cez Snap API. Môžem pracovať aj s reťazcom SVG. TODO

Snap podporuje animácie. Poskytuje jednoduché a intuitívne JavaScript API pre animáciu. Snap umožňuje urobiť SVG obsah viac interaktívnejší a zážívnejší. [13] TODO NIECO O DYNAMICKOM POUZITI A JSON PARSOVANI

2.2.4 SVG.JS

SVG.JS je ďalšia knižnica umožňujúca manipulovať a animovať SVG.

Medzi hlavné výhody knižnice patrí to, že je má ľahko čitateľnú syntax. Umožňuje animovanie veľkosti, pozície, transformácie, farby. Má modulárnu štruktúru, čo umožnuje používanie rôznych rozšírení. Existuje množstvo užitočných pluginou dostupných na internete. [14]

TODO NAPISAT NIECO PRECO SOM HO VYLUCILA

2.3 Zhodnotenie požiadaviek

Grafické komponenty sa budú vytvárať v programe Inkscape. Ovládanie a animovanie prostredníctvom knižnice Snap.svg.js. TODO PRESTYLIZOVAT.. Hlavný dôvod, prečo som sa

rozhodla pre túto knižnicu bol, že dokáže načítavať SVG súbor a potom s ním manipulovať.

Spĺňa požiadavku kompatibility pre moderné webové prehliadače. Je to open-source knižnica a má licenciu Apache 2.

Kapitola 3

Knižnica Snap.svg.js

3.1 append()

3.2 Element.attr(...)

Vráti alebo nastaví dané atribúty elementu.

3.2.1 Parametre:

- objekt - obsahuje pár kľúč-hodnota atribútov, ktoré chcem nastaviť.
- string - názov atribútu

Niekoľko možných dvojíc parametrov sú v tabuľke 3.1. Vráti buď súčasný element alebo stringovú hodnotu atribútu.

3.2.2 Použitie:

```
el.attr({
    fill: "#fc0",
    stroke: "#000",
    strokeWidth: 2,
```

```
});
```

3.3 Element.animate()

Snap.animation = function (attr, ms, easing, callback)

- attr (object) attributes of final destination - duration (number) duration of the animation, in milliseconds - easing (function) #optional one of easing functions of @mina or custom one - callback (function) #optional callback function that fires when animation ends

3.4 Gradinets, Path String, Colour Parsing

3.4.1 Paper.path([pathString])

Vytvorí <path> element podľa daného reťazca. Parameter pozostáva z jedno písmenkových príkazov, nasledovanými bodkami a oddeľovaný argumentami a číslami. Napríklad: ”M10,20L30,40 obsahuje príkazy: M s argumentami (10, 20) a L (30, 40). Rozdiel vo veľkosti písma vyjadruje to, či ide o absolútnu, alebo relatívnu cestu. Ak sú malé znaky jedná sa o relatívne, v prípade veľkých znakov absolútna cesta. Krátky zoznam príkazov je uvedený v tabuľke ....

3.5 Easing

- “linear”
- “¡” or “easeIn” or “ease-in”
- “¿” or “easeOut” or “ease-out”
- “¡¿” or “easeInOut” or “ease-in-out”
- “backIn” or “back-in”

Kľúč	Hodnota	Príklad použitia	Poznámka
cx	číslo	cx: 50	x-os súradnica centra kruhu, alebo elipsy
cy	číslo	cy: 90	y-os súradnica centra kruhu, alebo elipsy
r	číslo	r: 40	polomer kruhu, elipsy alebo okruhlých rohov na obdĺžniku
rx	číslo	r: 50	horizontálny polomer elipsy
ry	číslo	r: 40	vertikálny polomer elipsy
x, y	číslo	x: 50, y: 100	súradnica x-osi, y-osi
width, height	číslo	width: 500, height: 10	šírka, výška
”fill-opacity”	číslo	”fill-opacity”: 0.5	vyplnenie neprehľadnosti, nadobúda hodnoty od 0 po 1
fill	reťazec	fill: ”blue”	vyplnenie farbou, gradientom, obrázkom
stroke	reťazec	stroke: ”blue”	farba výplne okraja
strokeWidth	číslo	strokeWidth: 2	šírka okraja v px, default je 1
strokeLinecap	reťazec	strokeLinecap: ”round”	[”butt”, ”square”, ”round”]
strokeLinejoin	reťazec	strokeLinejoin: ”round”	[”bevel”, ”round”, ”miter”]
viewBox	pole		napr. viewBox: [0, 0, 800, 600]
strokeDasharray	reťazec	strokeDasharray: ”5 3”	pole čiarok, bodiek, pomlčiek
font	reťazec	font: ”20px Source Sans Pro, sans-serif”,	zmena písma, rodiny písma, veľkosti v pixeloch, a weight
transform	reťazec	transform: ”t”+ [0, 5] + ”r”+ 20	t - preloží sa na dané súradnice, r - otočí sa o daný úhol udaný v stupňoch
path	reťazec	path: ”M10,10 210,10”	SVG cesta
text	reťazec	text: ”snap”	zmení text elementu

Tabuľka 3.1: Výber možných parametrov pre funkciu Element.atrr(...)

Príkaz	**Názov**	**Parametre**
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizonal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+

Tabuľka 3.2: Niekoľko príkazov na tvorbu pathString

- “backOut” or “back-out”
- “elastic”
- “bounce”

Kapitola 4

Postup vytvorenia komponentov

4.1 UML diagram

Kapitola 5

Príklad vytvorenia grafického komponentu v Inkscape

5.1 Vytvorenie SVG v programe Inkscape

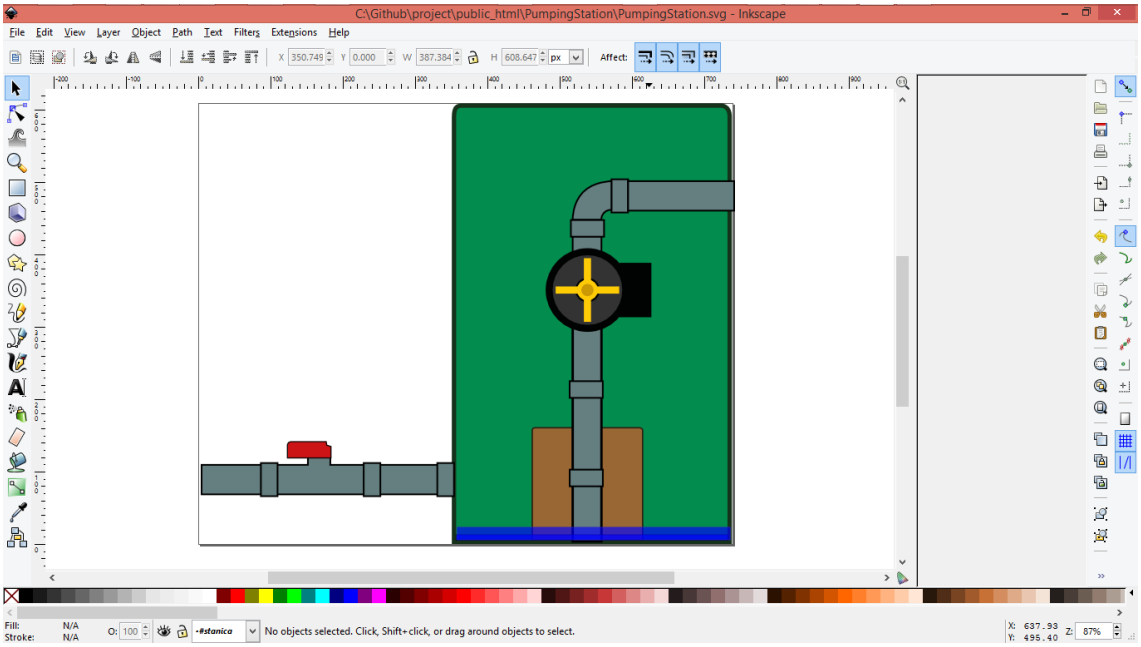
Nakreslenie jednotlivých častí komponentov prečerpávacej stanice bolo realizované pomocou ľavého bočného panela. Prečerpávacia stanica sa skladá z potrubí, indikátora úrovne hladiny vody, motora, a symbolu rotora čerpadla. Ako je možné vidieť na obrázku 5.1.

5.2 Definovanie id SVG

Pre ovládanie JavaScriptom je nutné si pozrieť jednotlivé jedinečné identifikačné názvy. Sú v SVG označované ako id. Zistenie id je pomerne jednoduché. Klikneme pravým tlačidlom na daný komponent, ktorého id chceme vedieť, a potom na Objekt Properties. Ako je možné vidieť na obrázku 5.2.

Po stlačení sa nám zobrazí nasledovné okno viď obrázok 5.3.

Z obrázka možno vyčítať aké je ID, predvolené sú tam napr. desc3072. Hodnoty je možné prepísať a zmeniť stlačením tlačidla Set. Pre nás je dôležitá hodnota v kolónke Label - #ventil. Toto nám umožni TODO TODO potom neskôr ako CSS selektor, cez ktorý budeme môcť ovládať danú časť. Spravidla hodnoty ID a Label sú rovnaké. Líšia sa iba v tom, že pri



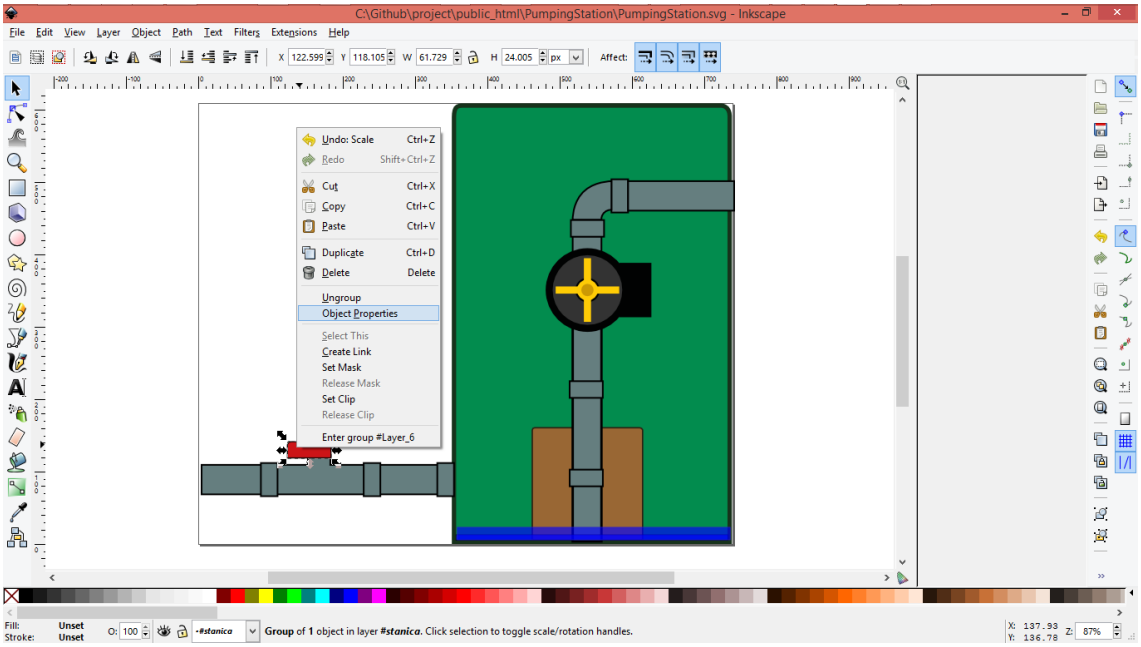
Obr. 5.1: Grafické prostredie programu Inkscape s nakreslenou prečerpávacou stanicou

Label je pridaný znak # pred názvom. ID je unikátny názov pre program Inkscape. TODO ROZDIEL MEDZI ID A LABEL DESCRIPTION A TITTLE... TODO Alebo ďalší spôsob zistenia id SVG časti tvarov je priamo nájsť tú hodnotu v PumpingStation.svg. Je to označené ako id=”ventil”.

V okne Object Properties je možné nastaviť script na animovanie. Po kliknutí na Interactivity sa zobrazia ďalšie kolónky, kde je možné zadať akciu, ktorá má nastať.

5.2.1 XML Editor

Ďalší spôsob získania informácii o svg cez Inkscape je cez zabudovaný XML Editor. Stlačením klávesovej skratky SHIFT + CTRL + X, alebo v hornej lište v menu vybrať ponuku Edit a na spodu je XML Editor. Následne sa zobrazí okno, ktoré je na obrázku 5.4.



Obr. 5.2: Zobrazenie menu pre daný objekt v Inkscape

5.3 Zistenie atribútov SVG

5.3.1 Prázdna nádrž

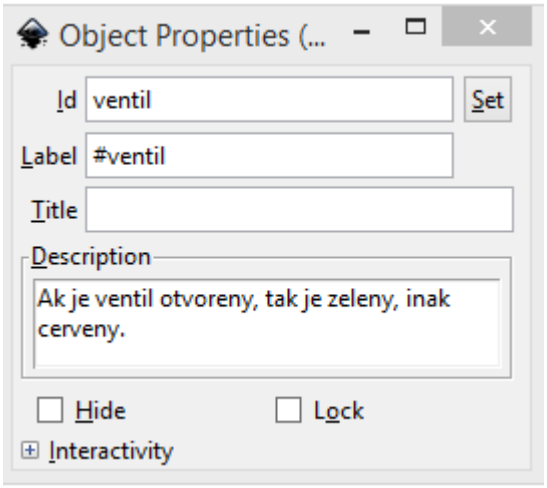
Prázdna nádrž je zobrazená na obrázku 5.5 Vyjadruje to, že do nádrže nevchádza tekutina. Hladina má výšku nastavenú na 9,64px. Šírka a súradnica x je rovnaká ako pri plnej nádrži. Súradnice osi y, x sú 1916,36 a 2507,84. Pri plnej nádrži sa zmení výška a os y. Pre animáciu zdvihnutia hladiny nádrže bude potrebné si zapamätať tieto súradnice.

5.3.2 Plná nádrž

Na obrázku č. 5.6 je vykreslená plná nádrž. Hladina má súradnice x, a šírku rovnakú ako v prázdnej nádrži. Zmenila sa os y, a výška - na 1320,16 a na 605,92.

V SVG súbore je to zapísané nasledovným kódom.

```
<rect
   inkscape:label="#hladina"
   y="1320.1689"
```

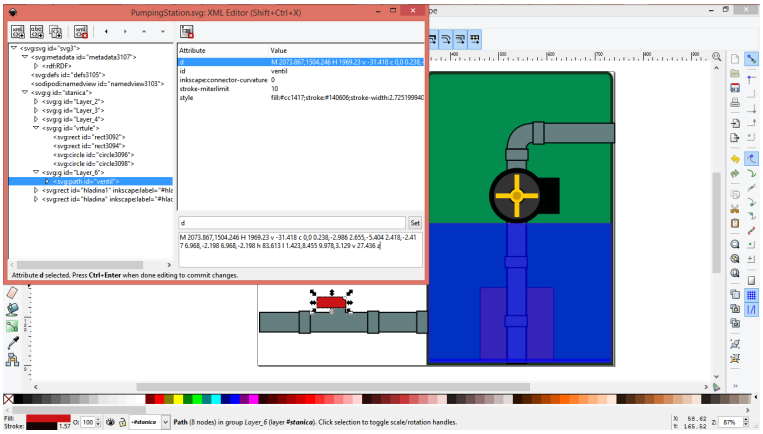


Obr. 5.3: Object Properties

```
   x="2507.8459"
   height="605.83868"
   width="797.04492"
   id="hladina"
   style=
     "fill:#0000ff;
␣␣␣␣fill-opacity:0.65098039;
␣␣␣␣fill-rule:evenodd;
␣␣␣␣stroke:#2c20c8;
␣␣␣␣stroke-width:7.42523718px;
␣␣␣␣stroke-linecap:butt;
␣␣␣␣stroke-linejoin:miter;
␣␣␣␣stroke-opacity:0.80952382">
</rect>
```

Parametre ako stroke, fill, a iné sa dajú meniť prostredníctvom FUNKCIE attr v Snap API.

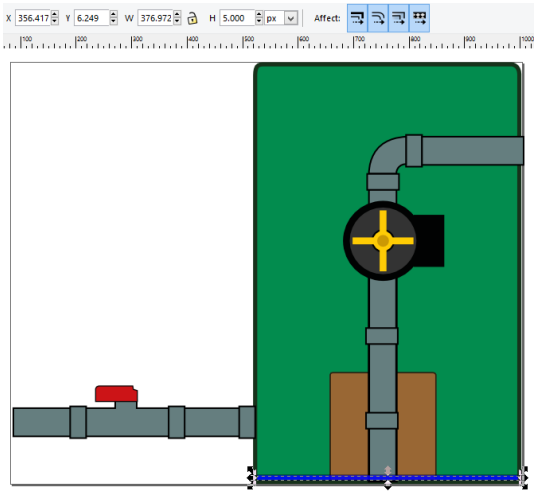
... TODO



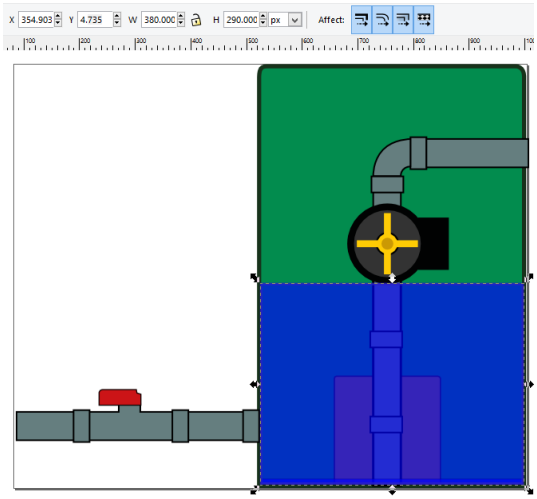
Obr. 5.4: Xml Editor v Inkscape

TODO POPIS SURADNICOVEHO SYSTEMU VEDIET VYSVETLIT SUVISLOST MEDZI DPI A PIX SURADNICAMI. .TODO

TODO MIERKY OBRAZKOV ZJEDNOTIT TODO aj ich lokalizaciu v dokumente TODO



Obr. 5.5: Prázdna nádrž prečerpávacej stanice



Obr. 5.6: Plná nádrž

Kapitola 6

Integrácia grafického komponentu pre dynamické ovládanie SVG objektu

TODO SUBOROVA STRUKTURA PRIDAT TODO

6.1 HTML súbor

Do HTML súboru index.html pridáme párový tag <svg>. Na toto miesto sa neskôr vykreslí SVG načítané zo súboru cez JavaScript. Môže sa tu uviesť i celý kód SVG obrázka. V prípade, že nebude v dokumente dané kde presne sa nachádza SVG tag tak sa pridá na najbližšie voľné miesto.

6.1.1 Kód

```
<svg
   id="svgStanica"
   viewBox="0 0 750 600"
   width="40%"
   height="40%">
</svg>
```

6.1.2 Vysvetlenie kódu

- **id** - jedinečný identifikátor, cez ktorý meníme vlastnosti.
- **viewBox** - je virtuálne okno, ktorým sa užívateľ uvidí svg obrázok. Je atribút, ktorý povoľuje špecifikovať danú množinu grafických komponentov, aby sa zobrazili v daných súradniciach x, y a šírke, výške. Hodnoty atribútov v viewBox sú štyri čísla - min-x, min-y, width a height.
- **width** a **height** je šírka a výška. Hodnoty atribútov je možné uviesť relatívne v percentách, alebo absolútne v pixloch.

Musíme sa uistiť, aby sa načítali všetky JavaScriptové knižnice, pred spustením funkcií. To zabezpečíme pridaním onload do tagu <body>.

```
<body onload="onPageLoad();">
```

A ešte jedna vec pri HTML súbore - TODO.

```
<script type="text/javascript" src="../js/snap.svg-min.js"
></script>
<script type="text/javascript" src="PumpingStation.js"></
script>
```

6.2 PumpingStation.js

TODO V súbore PumpingStation.js sú funkcie na animovanie.. TODO

6.2.1 onPageLoad()

Táto funkcia sa spustí pri načítaní tela HTML súboru index.html. Funkcia spustí funkciu PumpingStation(). Prvý parameter je udaný konkrétny svg súbor, ktorý chcem načítať. Druhý parameter je id tagu svg, ktorý je v html. TODO

```
function onPageLoad() {
  PumpingStation("PumpingStation.svg", "#svgStanica" );
}
```

6.2.2 PumpingStation(par1, par2)

Funkcia inicializuje daný svg súbor, a vykreslí ho. Parametre pre PumpingStation sú názov svg súboru, a id, ktoré sa nachádza v tagu <svg> html súbore.

```
var PumpingStation = function(nazovFileSVG, nameHTMLidSVG) {
  paper = Snap(nameHTMLidSVG);
  Snap.load(nazovFileSVG, function (f) {
    paper.append(f);
  });
};
```

paper - globálna premenná. TODO REFERENCIA NA PLOCHU ... Vytvorí plochu na kreslenie, alebo wraps existujúci SVG element. Ako parametre môžu byť buď šírka, výška, alebo DOM element. Napríklad Snap(600, 800), alebo Snap("#svgStanica"), resp Snap().

load - TODO funkciu z knižnice Snap. Cez ňu TODO načítam svg súbor. Ako parametre funkcie je názov súboru svg, prípadne môže byť prázdny, ak TODO nenačítavam súbor, ale beriem ho priamo z html súboru. Druhý parameter je funkcia, v ktorej TODO volám funkciu na zobrazenie obsahu svg súboru do daného tagu svg s id, ktorý bol daný pri funkcii paper. Na plochu TODO ho zobrazím pomocou príkazu append. TODO

6.2.3 Tank

Zanimovanie stúpania a klesania hladiny nádrže.

```
var Tank = {
  idTank: "#hladina",
  tank: function(){
    return  paper.select(this.idTank);},
```

```
  animateComponentTank: function(fillPerc) {
    if (fillPerc === undefined || fillPerc < 0) {
      fillPerc = 0;
    }
    var perHeight = 600 * (fillPerc / 100);
    var perY = 1912 - perHeight;
    this.tank().animate ( {
      height: perHeight,
      y: perY
    }, 800);
    return console.log("animacia tanku " + fillPerc);
  }
};
```

TODO PRESTYLIZOVAT VYHODIT RODY MOJE TODO TODO Vytvorila som objekt Tank medzi jeho atribúty patria: idTank, funkcia tank, a animateComponentTank. IdTank - je stringové - je to id, ktoré som získala zo svg súboru, alebo cez Inkscape ako Label. Funkcia tank - vyberie daný objekt, ktorý chcem ovládať. Pomocou Tank.tank() môžem volať funkcie z Snap knižnice.

Zanimovanie tanku je realizované v funkcii animateComponentTank - kde parametrom je v percentách udané o koľko sa ma zdvihnúť hladina nadrze. Využívam funkciu animate. Kde v prvom parametri - mením výšku a os y. Hodnotu perHeight je výška 600, ktorú vynásobím percentom o ktoré sa ma posunúť. PerY je hodnota, o ktorú sa posuniem po y-osi. Je vypočitaná ako 1912 co je y prázdnej nádrže a je od nej odpočítaná hodnota výšky. Ďalší parameter pri funkcii animate() je rýchlosť animácie vyjadrená v milisekundách.

6.2.4 Ventil

```
var Valve = {
  idValve: "#ventil",
  valve: function (){ return paper.select(this.idValve);},
```

```
  colorValve: "red",
  changeIsOpen: function (isOpened) {
    isOpened = (isOpened) ? 0 : 1;
    this.colorValve = (isOpened) ?  "red" : "green";
    this.valve().attr({fill: this.colorValve});
    return;);
  }
}
```

Farba sa dá zmeniť aj príkazom

`Valve.valve().attr({fill: \green"});.`

Názov farby môže byť uvedený slovne, alebo ako RGB.

Zmena farby Valve -

```
this.valve().attr  ({fill: this.colorValve});
```

6.3 TODO TRANSFORM

TODO TRANSFORM .. TODO

Kapitola 7

Návrh REST API

7.1 JSON

TODO PREROBIT CELE - TODO -

API k Pumping station schéme.

```
var updateData = {
  "valve": "true",
  "tank": "20",
  "engine": "20"
};
```

Tento kód definuje objekt s názvom updateData, ktorá má tri vlastnosti.

Interface funkcia k REST API.

```
function updateSchema(updateData){
  updateSchema01(updateData.valve, updateData.tank,
  updateData.engine);
}
```

Kapitola 8

Automatické mapovanie API

Analyzujte možnosti automatického mapovania API grafických prvkov pomocou metadát na existujúce API dostupné pre SCADA server D2000.

Kapitola 9

TODO — Analýza výkonnosti a obmedzení SVG

http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/

`http://caniuse.com/#feat=svg-html`

****

Animácia pozície - transform: translate(npx, npy);

Animácia škály - transform: scale(n);

Animácia otáčania - transform: rotate(ndeg)

Animácia neprehľadnosti - opacity: 0..1;

****

`http://www.svgopen.org/2008/papers/74-HighPerformance_GML_to_SVG_Transformation_for_the_Visual_Presentation_of_Geographic_Data_in_WebBased_Mapping_Systems/`

Transformation *scale*(sx, sy) - zmenim veľkosť tvaru na danane suradnice, *translate*(tx, ty) - presuniem na ine miesto - zmenim suradnice

`https://developers.google.com/web/fundamentals/performance/rendering/optimize-js?hl=en`

Podpora svg v prehliadačoch `http://caniuse.com/#feat=svg`

`http://www.schepers.cc/svg/blendups/embedding.html`

Záver

V mojej práci som sa snažila nájisť najjednoduchšie riešenie pre vizualizáciu komponentov. Vykreslovanie podľa súradnic, a priamo kreslenie cez JavaScript sa mi zdalo nepraktické z dôvodu, že priamo nevidím to čo kreslím. Preto som hľadala také riešenie, ktoré by mi umožňovalo ovládať už vytvorený obrázok ovládať cez JavaScript.

Cez Inkscape sa nakreslí komponent, a cez JavaScript pomocou knižnice Snap.svg.js sa manipuluje. Podarilo sa mi aj to, aby bol výsledný prvok responzívny aj na iných platformách ako napríklad tablety. SVG podporujú všetky moderné webové prehliadače.

Moje riešenie používa knižnicu a softvér, ktorý je open-source. Všetky zdrojové kódy práce sú v Git repository na GitHube.

Literatúra

[1] Dawber D., *Learning Raphael JS Vector Graphics* , Packt Publishing, 2013, ISBN 978-1-78216-916-1

[2] Wilson CH., *RaphaelJs Graphic and visualization on the web* , O'Reilly Media, 2013, ISBN 978-1-449-36536-3

[3] Haverbeke M., *Eloquent Javascript* 2. vyd. No Starch Press, 2014, ISBN 978-1-59327-584-6

[4] Zakas N. Z., *JavaScript pro webové vývojáře Programujeme profesionálně*, v1. vyd. Brno:Computer Press, a.s., 2009, ISBN 978-80-251-2509-0

[5] Suehring S., *JavaScript krok za krokem*, 1. vyd. Brno:Computer Press, 2008, ISBN 978-80-251-2241-9

[6] Zakas N. C., McPeak J., Fawcett J., *Profesionálně Ajax*, Zoner Press, 2007, ISBN 978-80-86815-77-0

[7] Eisenberg D. J., *SVG Essentials*, O'Reilly Media 2002, ISBN 978-0-596-00223-7, dostupné na `http://commons.oreilly.com/wiki/index.php/SVG_Essentials`

[8] Richardson L., Amundsen M., *RESTful Web APIs* 1. vyd. O'Reilly Media, 2013, ISBN 978-1-449-35806-8

[9] Allamaraju S., *RESTful Web Services Cookbook* 1. vyd. O'Reilly Media, 2010, ISBN 978-0-596-80168-7

[10] The JavaScript SVG library for the modern web, `http://snapsvg.io/`.

[11] http://raphaeljs.com/

[12] http://d3js.org/

[13] http://snapsvg.io/

[14] http://www.svgjs.com/

[15] Inkscape is a professional vector graphics editor for Windows, Mac OS X and Linux. It's free and open source. http://www.inkscape.org/en/about/features/

Zoznam skratiek

RGB Red Green Blue

XML EXtensible Markup Language

SVG Scalable Vector Graphics

JPEG Join Photographic Experts Group

GIF Graphics Interchange Format

SCADA Supervisory Control and Data Acquisition

HTML Hyper Text Markup Language

API Application Programming Interface

REST Representational State Transfer

JSON JavaScript Object Notation

W3C World Wide Web Consortium

DOM Document Object Model

CSS Cascading Style Sheets

D3 Data Driven Document

VML Vector Markup Language

WYSIWYG What You See Is What You Get

Zoznam termínov