VYŠŠÍ ODBORNÁ ŠKOLA A STŘEDNÍ ŠKOLA, s.r.o.

České Budějovice Pražská 3

ABSOLVENTSKÁ PRÁCE

2008 **Tereza Fuková**

Prohlašuji, že jsem absolventskou práci na téma TECHNOLOGIE SVG vypracovala samostatně a že jsem veškerou použitou literaturu uvedla v seznamu použitých zdrojů.

V Českých Budějovicích dne 16.5.2007

VYŠŠÍ ODBORNÁ ŠKOLA A STŘEDNÍ ŠKOLA, s.r.o.

České Budějovice Pražská 3

Studijní obor: **Výpočetní technika a programování**

Technologie SVG - aktuální standard webové vektorové grafiky

Absolventská práce

Autor: Tereza Fuková

Vedoucí absolventské práce: PaedDr. Petr Pexa

Tímto bych chtěla poděkovat PaedDr. Petru Pexovi za odborné vedení, připomínky a cenné rady při vypracování absolventské práce.

Obsah

1. Úvod

Technologie SVG je standard vektorového grafického formátu sloužící k definici dvojrozměrné grafiky, který je popsaný jazykem XML.

Tato technologie představuje široké možnosti a obrovské výhody týkající se hlavně oblasti webových stránek. Díky SVG se grafika na www stránkách bude zobrazovat správně i při různých velikostech a nebude docházet ke ztrátám na kvalitě. Dále je velkým přínosem malá velikost SVG souboru, která tak umožní rychlejší a pohodlnější načítaní www stránek.

Touto prací bych chtěla čtenářům přiblížit možnosti a výhody SVG a ukázat, jak se grafika pomocí této technologie vytváří.

V úvodní části práce se budu zabývat hlavními rozdíly mezi vektorovou a bitmapovou grafikou a také základními informacemi o jazyku XML.

V hlavní části se zaměřím na historii a základní informace o SVG. Také se budu zabývat vytvořením grafických primitiv, jejich vlastnostmi, různými efekty a v poslední řadě i animací.

Na závěr bych se ráda zabývala některými editory SVG a v poslední řadě bych ráda zmínila podporou SVG v prohlížečích.

2. Bitmapová a vektorová grafika

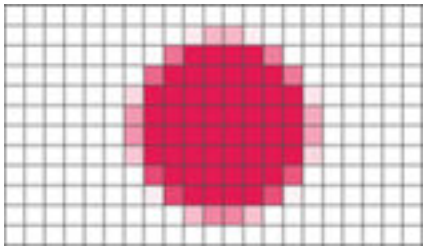
2.1. Bitmapová (rastrová) grafika

2.1.1. Definice

Rastrová grafika představuje jeden ze dvou základních způsobů, jak ukládat obrázky.

V bitmapové grafice se celý obrázek skládá z jednotlivých bodů uspořádaných do mřížky (rastru), tzv. bitmapy, kde je u každého bodu definována barva a jas. U obrázků s barevným modelem RGB (R – red, G – green, B - blue) má každý pixel alespoň tři bajty. Čím více možných barevných tónů, tím bude informace o každém bodě datově objemnější. U černobílé grafiky stačí každému pixelu pouze jeden bit.

Každá bitmapová grafika musí mít danou velikost (počet pixelů vertikálně a horizontálně) a barevnou hloubku (počet bitů na pixel).



Obr. č. 1 – Bitmapová grafika

2.1.2. Výhody

Výhodou rastrové grafiky je její veliká podpora a snadné pořízení (např. pomocí fotografie nebo skeneru). Mezi základní formáty, které lze dnes otevřít skoro na každém počítači, řadíme BMP, GIF, TIF, PNG a JPEG. Na bitmapovou grafiku lze rovněž aplikovat rozsáhlé množství různých efektů, na rozdíl od vektorové grafiky, pro kterou jich není mnoho.

2.1.3. Nevýhody

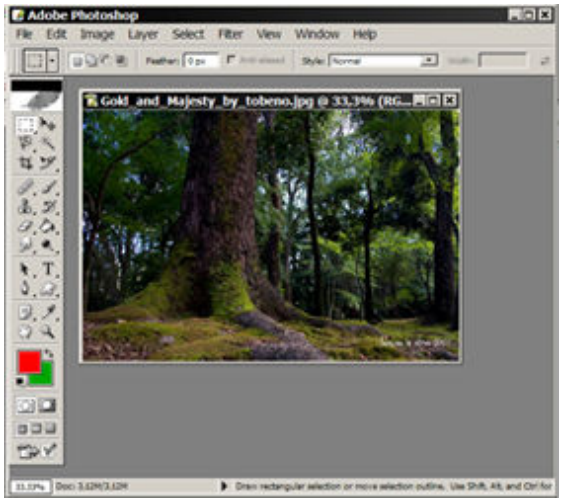
Kvůli skutečnosti, že každý pixel musí nést informaci o barvě a jasu, je její hlavní nevýhoda velká datová náročnost. Další velkou nevýhodou je, že ji nelze bez snížení kvality zvětšovat, protože dochází k roztahování a vyhlazování pixelů.

2.1.4. Využití

Bitmapová grafika má veliké uplatnění například u fotografií nebo složitých ilustrací plných stínů a rozmanitých barev a vyniká napříč všemi počítačovými obory. Uplatní se v drobných grafických prvcích na internetových stránkách, bitmapových texturách aplikovaných na 3D objekty či fotografiích připravených pro DTP.

V současné době je nejpoužívanějším programem Adobe Photoshop. Jeho nativní formát PSD umožňuje ukládání rastrové grafiky ve vrstvách spolu s vektorovými objekty i textem.

Pro kvalitní přenos fotografií se často používá formát TIF, který je ovšem pro svou datovou náročnost nevhodný pro použití na webu nebo v digitálních fotoaparátech. Nejrozšířenějším formátem na webu je JPEG nebo GIF.



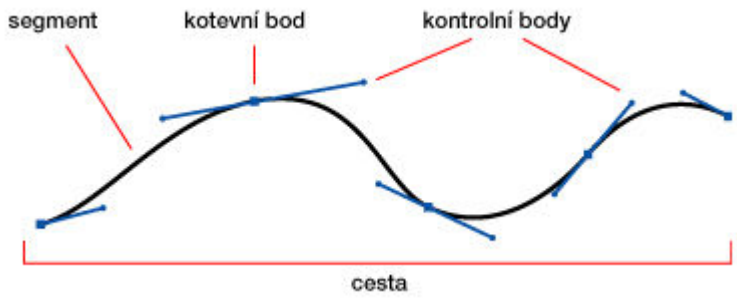
Obr. č. 2 – Adobe Photoshop

2.2. *Vektorová grafika*

2.2.1. Definice

Vektorová grafika představuje spolu s rastrovou grafikou dva základní způsoby, jak ukládat obrázky.

Obrázky vektorové grafiky jsou vytvářeny pomocí vektorů (křivek). Francouzský matematik Sierr Béziér vyvinul metodu, pomocí níž lze popsat libovolný úsek dané křivky pouze čtyřmi body. Je nutné znát dva krajní tzv. kotevní body a dva tzv. kontrolní body, které určují vlastní tvar křivky. Spojnicí mezi kontrolním a kotevním bodem vznikne tečna k výsledné křivce. Křivka může být otevřená nebo zavřená, s výplní či bez ní.



Obr. č. 3 - Béziérova křivka

2.2.2. Výhody

Hlavní výhodou oproti bitmapové grafice je libovolné zvětšování obrázku bez ztráty na kvalitě. Dále nám vektorová grafika umožňuje pracovat s každým objektem v obrázku odděleně. Také výsledná velikost obrázku je o něco menší než u bitmapové grafiky.

2.2.3. Nevýhody

Mezi nevýhody vektorové grafiky patří složitější pořízení obrázku, který naopak u bitmapové grafiky pořídíme snadno pomocí fotografie. Pokročí-li složitost grafického objektu, začíná být vektorová grafika náročná na paměť, procesor a velikost disku.

2.2.4. Využití

Vektorová grafika je zejména využívána pro tvorbu log, diagramů, počítačovou sazbu nebo tvorbu jednoduchých ilustrací. Významnou roli také hraje vektorová grafika při animacích v oblíbeném prostředí Macromedia Flash™.

Mezi nejpoužívanější editory vektorové grafiky řadíme Adobe Illustrator s jeho nativním formátem AI nebo Corel Draw s formátem CDR. Univerzálním souborem pro vektorovou grafiku je například EPS (Encapsulated PostScript), který byl vytvořen pro přenos obrazových dat určených pro tisk. Dalším populárním formátem vektorové grafiky je PDF (Portable Document Format) a v neposlední řadě formát SVG (Scalable Vector Graphics).

Obr. č. 4 - Ukázka efektivity vektorové grafiky při zvětšování

(a) originální vektorový obrázek

(b) zvětšeno jako vektorový obrázek

(c) zvětšeno jako rastrový obrázek.

3. XML

3.1. Definice

XML (eXtensible Markup Language) je značkovací jazyk vyvinutý a standardizovaný konsorciem W3C (World Wide Web Consortium). Jazyk XML, který umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, slouží především pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

3.2. Vlastnosti

XML je otevřený formát založen na jednoduchém textu a je zpracovatelný libovolným textovým editorem. Implicitní znakovou sadou se používá Unicode (ISO 10646), která nám umožňuje vytvářet dokumenty obsahující texty ve více jazycích.

Vzhled vytvořeného XML dokumentu je potřeba definovat pomocí stylu. Mezi nejpoužívanější stylové jazyky řadíme CSS (Cascading Style Sheets), XSL (eXtensible Stylesheet Language) a DSSSL (Document Style Semantics and Specification Language).

Dokumenty XML se musí řídit určitými pravidly, které lze definovat v DTD (Document Type Definition). Poté lze automaticky provádět kontroly tzv. parserem, zda vytvořený XML dokument odpovídá dané definici.

Mezi nejběžnější DTD patří:

- XHTML (Extensible Hyper Text Markup Language) – nástupce jazyka HTML
- RDF (Resource Description Framework) – specifikace, kterou je možno využít pro přidávání metainformací k datovým zdrojům
- RSS – slouží pro čtení novinek na webových stránkách
- SMIL (Synchronized Multimedia Integration Language) – slouží pro tvorbu multimediálních prezentací pomocí jazyka XML

- MathML (Mathematical Markup Language) – jazyk umožňující zápis matematických výrazů
- OSD (Open Software Description Format) – pro popis softwarových aplikací a jejich částí
- PGML(Precision Graphics Markup Language) - návrh jazyka určeného pro zařazování dvourozměrné vektorové grafiky na webové stránky
- UXF (UML eXchange Format) – pro výměnu dat v jazyce UML
- SVG (Scalable Vector Graphics) – formát pro dvourozměrnou grafiku určený hlavně pro webové stránky
- TEI (Text Encoding Initiative) – rozsáhlý projekt, jehož cílem je vytvořit skupiny DTD vhodných pro uchovávání a výměnu knih

XML také umožňuje vytvářet odkazy v rámci dokumentu nebo mezi jednotlivými dokumenty.

Tvorbu odkazů popisují tři standardy:

- Pointer (XML Pointer Language) – používá s k určení jednotlivých části v dokumentu
- XPath (XML Path Language) – umožňuje adresovat jednotlivé části dokumentu
- XLink (XML Linking Language) – odkazy na jednotlivé dokumenty určuje podle jejich URL adresy

3.3. Syntaxe

Dokument XML, který splňuje daná pravidla se nazývá správně strukturovaný (well-formed).

Mezi základní požadavky řadíme:

1) Celý dokument musí být uzavřen mezi počáteční a ukončovací tag.
2) Pro každý počáteční tag <tag> musí existovat tag ukončovací </tag>. Výjimku tvoří prázdné elementy (například
).
3) Všechny hodnoty atributů musí být uzavřeny v uvozovkách nebo apostrofech.
4) Jména všech elementů a atributů jsou citlivá na velikost písmen.
5) Elementy mohou být do sebe vnořeny, ale nemohou se překrývat.
6) Pokud v dokumentu používáme jiné kódování než UTF-8, je to nutné uvést v XML deklaraci:

```
<?xml version="1.0" encoding="iso-8859-2"?>
```

Jednoduchý seznam v XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<seznam>
  <titulek>Seznam filmů</titulek>
  <film>
      <nazev>Marečku, podejte mi pero!</nazev>
      <zanr>Komedie</zanr>
      <cislodvd>1</cislodvd>
  </film>
  <film>
      <nazev>Láska nebeská</nazev>
      <zanr>Romantický</zanr>
      <cislodvd>2</cislodvd>
  </film>
  <film>
      <nazev>Vymítač ďábla</nazev>
      <zanr>Horor</zanr>
      <cislodvd>3</cislodvd>
  </film>
</seznam>
```

4. Technologie SVG

4.1. Základní informace o SVG

SVG (Scalable Vector Graphics – škálovatelná vektorová grafika) je značkovací jazyk navržený hlavně pro oblast webové grafiky, popisující dvojrozměrnou vektorovou grafiku pomocí XML.

Scalable Vector Graphics:

- Scalable - zdůrazňuje možnost zvětšování či zmenšování bez ztráty na kvalitě obrázků nebo tisku s vysokým rozlišením
- Vector Graphics – vektorová grafika umožňuje téměř neomezené možnosti

SVG slouží nejen k zobrazení statických či animovaných obrázků, ale umožňuje i vybudování mapových portálů, geografických informačních systémů či jednodušších her.

Grafické schopnosti SVG:

1) SVG definuje tři základní typy grafických objektů:
   - vektorové tvary (vector graphics shapes) – obdélník, kružnice, elipsa, úsečka, lomená čára, mnohoúhelník, čára
   - rastrové obrazy (raster images)
   - textové objekty
2) Objekty jsou vykreslovány ve stejném pořadí, ve kterém jsou zapsány do zdrojového kódu.
3) Na libovolný grafický element lze aplikovat různé bitmapové efekty a daný element přitom zůstává ve vektorové podobě a teprve po vykreslení prohlížečem se daný efekt projeví.
4) Textové objekty zůstávají stále textem s možností fulltextového vyhledávání, kopírování vybraného textu do editoru či zachování typografie.

4.2. Historie a vývoj SVG

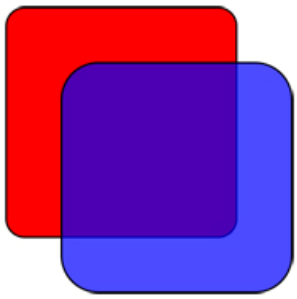
Vývoj SVG započal v roce 1998 a byl inspirován staršími jazyky PGML a VML. SVG odpovídá internetovým standardům v organizaci W3C a v roce 2001 vyšla finální norma W3C Recomendation SVG 1.0.

V roce 2003 vzniká norma SVG 1.1, která víceméně pouze rozděluje specifikace SVG 1.0 na menší části v zájmu implementace SVG na méně výkonná zařízení. Vzniká profil SVG Tiny (pro mobilní telefony), ze kterého je vypuštěna podpora CSS stylů, filtrů, skriptů, gradientů, vzorků a průhlednosti. Druhý profil SVG Basic je určen pro zařízení typu PDA. Z původní specifikace jsou omezeny jen některé filtry.

V roce 2005 vzniká specifikace SVG 1.2, která přichází s dalším vylepšením týkajícím se především typografických schopností, možnosti práce s multimediálním obsahem či interaktivity.

4.3. Ukázka SVG

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
    width="467" height="462">
  <rect x="80" y="60" width="250" height="250" rx="20"
      fill="red" stroke="black" stroke-width="2px" />
  <rect x="140" y="120" width="250" height="250" rx="40"
      fill="blue" fill-opacity="0.7" stroke="black" stroke-
width="2px" />
</svg>
```



Obr. č. 5 – obrázek k danému kódu

5. Kurz SVG

5.1. SVG dokument

5.1.1. Hlavička dokumentu

Stejně jako každý jiný XML dokumentu i SVG dokument musí obsahovat na začátku hlavičku obsahující informace o verzi XML, kódování a DTD.

Ukázka:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

5.1.2. Tělo dokumentu SVG

Za hlavičkou dokumentu následuje hlavní část - tělo, začínající elementem `<svg></svg>`. Tento element může obsahovat atributy uvádějící velikost obrázku, popřípadě atribut `xmlns`, `version` nebo `viewBox` (`viewBox=(x y šířka výška)` – umožní definovat virtuální okno). Dále následují vlastní příkazy pro vytvoření grafiky.

Ukázka:

```
<svg width="" height="" viewBox="" version="1.1"
      xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
</svg>
```

5.2. Grafická primitiva

Grafickými primitivy se rozumí základní vektorové objekty. Těmto objektům lze přiřadit některé ze základních atributů jako jsou například barva výplně `fill`, barva obrysů `stroke` nebo šířka obrysů `stroke-width`.

Mezi grafická primitiva řadíme:

- obdélník – `rectangle`
- kruh – `circle`
- elipsa – `ellipse`

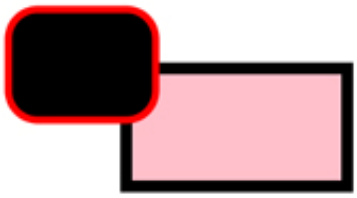
- úsečka – `line`
- polyčára – `polyline`
- mnohoúhelník – `polygon`
- cesta – `path`

5.2.1. Obdélník

Obdélník je zapisován pomocí `<rect />` a měl by obsahovat minimálně čtyři základní atributy - `x`, `y` (pro určení levého horního bodu) a `width a height` (pro specifikaci rozměrů). Pokud chceme u obdélníku zaoblené rohy, použijeme atributy `rx` a `ry` určující poloměr os elipsy, kterými jsou následně nahrazeny všechny čtyři rohy.

Ukázka:

```
<svg width="300" height="150" viewBox="0 0 250 200"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
     <rect x="90" y="50" width="150" height="80" fill="pink"
           stroke="black" stroke-width="8" />
     <rect x="50" y="10" width="100" height="75" rx="20"
           stroke="red" stroke-width="5" fill="black" />
</svg>
```



Obr. č. 6 - obrázek k danému kódu

5.2.2. Kruh

Kruh je zapisován pomocí `<circle />` a měl by obsahovat minimálně tři atributy – `cx`, `cy` (pro určení souřadnic středu kružnice) a `r` (poloměr).

Ukázka:

```
<svg width="300" height="150" viewBox="0 0 300 180"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
     <circle cx="90" cy="90" r="50" fill="blue" />
     <circle cx="120" cy="120" r="30" />
</svg>
```

Obr. č. 7 - obrázek k danému kódu

5.2.3. Elipsa

Elipsa se značí elementem `<ellipse />` a měla by mít nastavené čtyři již známé atributy `cx`, `cy` a `rx`, `ry`.

5.2.4. Úsečka

Úsečka se zapíše pomocí `<line />` a určíme ji pomocí atributů `x1`, `y1`, `x2` a `y2`, které určí souřadnice obou vrcholů úsečky.

Ukázka:

```
<svg width="200" height="200" version="1.1"
     xmlns="http://www.w3.org/2000/svg">
     <line stroke="red" stroke-width="4" x1="10" y1="20" x2="150"
      y2="20" />
     <line stroke="red" stroke-width="4" x1="50" y1="40" x2="150"
      y2="40" />
     <line stroke="black" stroke-width="4" x1="20" y1="10" x2="20"
      y2="100" />
     <line stroke="red" stroke-width="4" x1="40" y1="30" x2="150"
      y2="30" />
</svg>
```

Obr. č. 8 - obrázek k danému kódu

5.2.5. Polyčára

Polyčára je geometrický tvar složený z libovolného počtu na sebe navazujících úseček. Zapisuje se pomocí `<polyline />` a má pouze jeden atribut `points`, do kterého se zapisují hodnoty určující souřadnice vrcholů úseček oddělených čárkou. Jako jediný tvar nemůže být vyplněn, a to ani v případě, že koncový bod poslední úsečky je shodný s počátečním bodem polyčáry.

5.2.6. Mnohoúhelník

Mnohoúhelník se zapisuje pomocí elementu `<polygon />` a k jeho určení je potřebný pouze jeden atribut `points`. Na rozdíl od polyčáry může být ovšem vyplněný.

Ukázka:

```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
     <polygon fill="orange" stroke="red" stroke-width="3"
      points="100,150 150,100 200,150, 200,250" />
</svg>
```

Obr. č. 9 - obrázek k danému kódu

5.2.7. Cesty

Cesty jsou určeny pro obecné kreslení a zapisují se pomocí elementu `<path />`.

Nejdůležitější atributy:

- `d` – pro vlastní geometrii cesty
- `class` – třída
- `style` – styl, kterým má být cesta vykreslena
- `id` – identifikace cesty
- `transform` – lineární transformace aplikovaná na všechny specifikované vrcholy

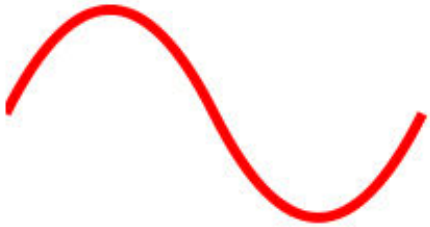
Do atributu `d` zapisujeme příkazy, které jsou dané jedním písmenem a potřebnými souřadnicemi. Písmena lze zapsat dvěma způsoby. Pokud napíšeme malé písmeno jedná se o relativní pohyb a pokud velké písmeno jde o pohyb absolutní.

K dispozici máme následující příkazy:

1) Moveto (`m`) – přesun pera bez kreslení na dané souřadnice
2) Lineto (`l`) – vykreslení úsečky z dané polohy pera na nové souřadnice
3) Lineto (`h`) – vykreslení horizontální úsečky
4) Lineto (`v`) – vykreslení vertikální úsečky
5) Closepath (`z`) – uzavření cesty
6) Curveto (`c`) – kubická Beziérova křivka
7) Quadratic Beziére (`q`) – kvadratická Beziérova křivka

Ukázka:

```
<svg width="230" height="130" version="1.1" viewBox="0 0 200 100
"xmlns="http://www.w3.org/2000/svg" >
      <path stroke="red" stroke-width="5" fill="none" d="M 0 100 Q
      50 0 100 100 T 200 100" />
</svg>
```



Obr. č. 10 - obrázek k danému kódu

5.3. *Text*

V SVG lze mimo jiné také pracovat s textovými objekty, které lze různě modifikovat. Nemusíme se omezovat pouze na jednořádkový text, ale je možné vytvořit i text odstavcový. Na text lze aplikovat různé výplně či přechody nebo lze upravovat barvu, styl či šířku obrysu. Jako standardní kódování textu je doporučeno UTF-8.

5.3.1. Element `text`

Základním prvkem pro vkládání textu do souborů je element `<text>`.

Atributy elementu `text`:

- `x`, `y` – poloha textu
- `font-family` – nastavuje písmo podle pravidel v CSS
- `font-style` – nastavuje styl písma (např.: kurzíva - `italic`)
- `font-weight` – nastavuje tloušťku písma
- `font-stretch` – nastavuje vodorovnou deformaci písma
- `font-size` – nastavuje velikost písma
- `kerning` – nastavuje mezery mezi písmeny, lze nastavit konstantní hodnotu nebo hodnotu `auto`
- `letter-spacing` – nastavuje mezery mezi písmeny, které jsou připočteny k automatickému kerningu

- `word-spacing` – nastavuje mezery mezi slovy
- `text-decoration` – nastavuje styl písma obdobně jako v CSS (např.: `underline, overline, blink, none`)
- `text-anchor` – vodorovné zarovnání jednoho textového bloku (hodnoty: `start`, `middle`, `end`, `inherit` – dědičné, přebírá se hodnota od nadřazených prvků)
- `baseline-shift` – svislý posun znaků (hodnoty: `baseline`, `sub`, `super`, `inherit`)

5.3.2. Element `tspan`

U SVG je třeba zdůraznit základní rozdíl s prácí s textem v porovnání s XHTML. U SVG neexistuje automatické přelévání textu v rámci více řádek. Pro efekt odstavců je tedy nutné určit každý řádek pomocí vnořeného elementu `<tspan>`.

Atributy elementu `tspan`:

- `x`, `y` – poloha textu
- `dx` – atribut udává relativní posun znaků uvnitř textového prvku, lze uvést i více hodnot, které pak odpovídají ručnímu odsazení (kerningu) mezi písmenem na pozici `x` a `x-1`
- `dy` – tento atribut posouvá znaky nahoru či dolů
- `rotate` – pootočení každého znaku v aktuálním textovém prvku, opět může obsahovat více hodnot

Ukázka:

```
<svg width="350" height="200" viewBox="0 0 690 400"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
     <text font-family="Helvetica" font-size="45" x="150" y="150"
      fill="blue" > SVG
            <tspan dx="5" dy="50" font-weight="bold" fill="red"
             font-style="italic"> práce s </tspan>
            <tspan dx="30 35 50 65 80 100" fill="none"
             stroke="black" font-size="60" rotate="0 20 35 50 60
             75" > textem </tspan>
</text></svg>
```

Obr. č. 11 - obrázek k danému kódu

5.3.3. Element `tref`

Obsah elementu `<text>` může být tvořen buď přímo vloženými znaky, nebo nepřímo odkazem pomocí elementu `<tref />`.

Atributy elementu `tref`:

- `xlink:href` – odkaz na pojmenovaný prvek obsahující text

Ukázka:

```
<svg width="350" height="100" viewBox="0 0 200 150"
     xmlns="http://www.w3.org/2000/svg" version="1.1"
     xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
      <text id="ReferencedText">
      Text vložený nepřímo odkazem...
      </text>
</defs>
      <text x="50" y="50" font-size="20" fill="red" >
      <tref xlink:href="#ReferencedText"/>
      </text>
</svg>
```

5.4. Text na křivce

Text, který má být zobrazen na křivce se musí uzavřít do elementu `<textPath>` s atributem `xlink:href`, který odkazuje na prvek `<path />` s definicí křivky.

Pro eliminaci drobných rozdílů mezi prohlížeči, které mohou nastat při zobrazování kvůli složitějšímu výpočtu délky křivky, se může použít alternativní atribut `pathLength`.

5.4.1. Element `textPath`

Atributy elementu `textPath`:

- `xlink:href` – odkaz na prvek `path` obsahující křivku
- `startOffset` – nastaví aktuální polohu textu jako posun měřený od počátku křivky

Ukázka:

```
<svg width="600" height="300" viewBox="0 -50 1000 500"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
      <path id="krivka"
            d="M 100 200
            C 200 100 300 0 400 100
            C 500 200 600 300 700 200
            C 800 100 900 100 900 100" />
</defs>
<text font-family="AdineKirnberg-Script" font-size="120"
      fill="black" >
<textPath xlink:href="#krivka">
      Lorem
      <tspan fill="green" >
      ipsum
      </tspan>
      <tspan>
      dolor sit amet, consectet
      </tspan>
</textPath>
</text>
</svg>
```

Obr. č. 12 - obrázek k danému kódu

5.5. *Vlastnosti základních geometrických tvarů*

Základním geometrickým tvarům je možné přiřadit několik vlastností jako jsou například šířka, barva a styl obrysu, způsob ukončení křivky nebo barva a styl výplně. Pro nastavení stejných vlastností u více grafických objektů se používá párový element `<g>`. Všechny objekty uzavřené v tomto elementu mají vlastnosti nastavené jako atributy prvku `<g>`.

Základní atributy pro geometrické tvary:

1) `fill` – nastavuje barvu (implicitně nastaveno na `black`) výplně a může nabývat těchto hodnot:
   - `none` – žádná
   - `#rgb` – barvy definované pomocí RGB nebo pojmenované barvy
   - `inherit` – zděděná hodnota
2) `stroke` – nastavuje barvu úsečky nebo křivky a nabývá stejných hodnot jako atribut `fill`
3) `fill-opacity` – průhlednost výplně, nabývá hodnot od 0.0 (plně průhledná) do 1.0 (zcela neprůhledná)
4) `fill-rule` – vytvoří průhledné otvory v ploše, kde se překrývají vektorové prvky `path` (hodnoty: `nonzero` nebo `evenodd`)
5) `stroke-width` – šířka obrysů

6) `stroke-linecap` – styl ukončení:
   - `butt` – ukončení cesty kolmým řezem v místě koncových bodů
   - `round` – ukončení zaoblením
   - `square` - ukončení cesty kolmým řezem vzdáleným od koncových bodů
7) `stroke-linejoin` – styl napojení:
   - `miter` – okraje úseček jsou protaženy do místa svého protnutí
   - `round` – je vykreslené obloukové koleno
   - `bevel` – napojení je ukončené rovným okrajem
8) `stroke-opacity` – průhlednost úsečky nebo křivky
9) `stroke-dasharray` – typ čar (čárkované, tečkované atd.) zadávaný číselnými hodnotami (hodnoty na lichých místech udávají délku čar a hodnoty na sudých místech značí délku mezi čarami) oddělenými čárkami
10) `stroke-dashoffset` – definuje posun vzorů čárkování vůči skutečnému počátku tvaru
11) `stroke-mitterlimit` – omezí průřez v bodě spojení dvou čar
12) `display` – způsobí absolutní vyřazení prvku ze zobrazování (hodnoty: `none` a `inline`)
13) `visibility` – skrytí prvků (hodnoty: `hidden` a `visible`)

5.5.1. Prvek `Marker`

Na začátek či konec otevřených křivek je možné připojit určitý symbol, který se vytvoří pomocí párového elementu `<marker>`.

Atributy elementu `marker`:

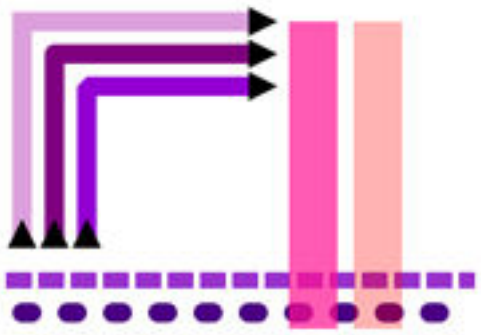
1) `id` – jednoznačný identifikátor použitý pro připojení symbolu ke geometrickému tvaru

2) `refX` – relativní poloha počátku symbolu na ose `x`
3) `refY` - relativní poloha počátku symbolu na ose `y`
4) `markerUnits` – jednotky použité při vytváření symbolu, výchozí hodnotou je:
   - `srokeWidth` – velikost symbolu se mění současně s velikostí tvaru
   - `userSpaceOnUse` – velikost symbolu je zadaná přímo v souřadnicích
5) `markerWidth` – šířka značky
6) `markeHeight` – výška značky
7) `orient` – způsob orientace symbolu vůči křivce:
   - `auto` – symbol se orientuje ve směru osy křivky
8) `viewBox` - rozsah hodnot souřadnic, které omezují symbol ze všech čtyř stran
9) `marker-start` – definuje symbol pro počáteční bod křivky
10) `marker-end` – definuje symbol pro poslední bod křivky
11) `marker-mid` – definuje symboly vykreslující se nad vnitřními body křivky

Ukázka s použitými vlastnostmi:

```
<svg width="300" height="300" s
      xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
<defs><marker id="sipka" viewBox="0 0 10 10" refX="0" refY="5"
       markerUnits="strokeWidth" markerWidth="1.5"
       markerHeight="1.5" orient="auto">
<path d="M 0 0 L 10 5 L 0 10 z" />
      </marker></defs>
      <polyline fill="none" stroke="plum" stroke-width="6"
            points="10 180 10 110 80 110" stroke-linejoin="miter"
            marker-start="url(#sipka)" marker-end="url(#sipka)" />
      <polyline fill="none" stroke="purple" stroke-width="6"
            points="20 180 20 120 80 120" stroke-linejoin="round"
            marker-start="url(#sipka)" marker-end="url(#sipka)"/>
      <polyline fill="none" stroke="darkviolet" stroke-width="6"
            points="30 180 30 130 80 130" stroke-linejoin="bevel"
            marker-start="url(#sipka)" marker-end="url(#sipka)" />
```

```
      <line stroke="darkorchid" stroke-width="5" stroke-
            dasharray="8,2" x1="5" y1="190" x2="150" y2="190" />
      <line stroke="indigo" stroke-width="6" stroke-
            linecap="round" stroke-dasharray="3,11" x1="10"
            y1="200" x2="150" y2="200" />
      <line stroke="deeppink" stroke-opacity="0.7" stroke-
            width="15" x1="100" y1="110" x2="100" y2="205" />
      <line stroke="red" stroke-opacity="0.3" stroke-width="15"
            x1="120" y1="110" x2="120" y2="205" />
</svg> >
```



Obr. č. 13 - obrázek k danému kódu

5.5.2. Gradientní výplně

V grafickém formátu SVG je možné použít dva typy výplní s přechodem (gradientní výplně), lineární přechod nebo radiální přechod. Tyto přechody se definují mezi párový element `<defs>`, který se používá dále také pro definici symbolů, vzorů výplně či typu čar.

Pro vytvoření lineárního nebo radiálního přechodu se používají párové elementy `<linearGradient>` a `<radialGradient>`. Při definici přechodů se nemusíme omezovat pouze na dvě barvy, ale lze vytvořit několik úseků pomocí prvku `<stop />`, ve kterém následně definujeme lokalizaci, barvu či průhlednost.

Barevné prostory:

- `sRGB` – nelineární prostor s křivkou gama korekce 2.2
- `linearRGB` - lineární průběh všech barevných kanálů bez gama korekce

Atributy elementu `linearGradient` a `radialGradient`:

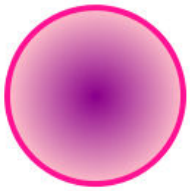
1) `id` – jednoznačný identifikátor
2) `color-interpolation` – způsob přechodu mezi dvěma barvami (hodnoty: `auto`, `sRGB` – výchozí hodnota, `linearRGB`, `inherit`)
3) `color-interpolation-filters` – filtry (hodnoty: `auto`, `sRGB`, `linearRGB` – výchozí hodnota, `inherit`)
4) `gradientUnits` – vybírá jeden ze dvou možných systémů pro udání atributů definujících vektor (u `linearGradient`) nebo kružnici (u `radialGradient`) přechodu
   - `userSpaceOnUse` – použije se aktuální systém souřadnic právě vyplňovaného objektu
   - `objectBoundingBox` - definuje souřadnicový systém orientovaný uvnitř rámečku ohraničujícího konkrétní objekt
5) `gradientTransform` – dodatečné souřadnicové transformace aplikující se na přechod
6) `spreadMethod` – definuje opakování určitého přechodu (hodnoty: `pad` – jedno opakování, `reflect` - zrcadlení, `repeat` – nekonečné opakování)
7) `xlink:href` – odkaz na jiný přechod
8) `x1, y1, x2, y2` – souřadnice určující úsečku, podél které se přechod vykresluje (u `linearGradient`)
9) `cx, cy, r` – vnější kružnici v přechodu (u `radialGradient`)
10) `fx, fy` – středový bod přechodu (u `radialGradient`)

Atributy elementu `stop`:

- `offset` – definuje umístění bodu na úsečce mezi dvěma sousedícími body přechodu v rozsahu 0 až 1
- `stop-opacity` – definuje průhlednost
- `stop-color` – definuje barvu

Ukázka:

```
<svg width="300" height="150" viewBox="0 0 200 150"
      xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
<defs><radialGradient id="Gradient"
      gradientUnits="objectBoundingBox" cx="50%" cy="50%" r="50%">
      <stop offset="0%" stop-color="darkmagenta" />
      <stop offset="100%" stop-color="pink" />
      </radialGradient>
</defs>
<circle cx="40" cy="40" r="30" fill="url(#Gradient)"
stroke="deeppink" stroke-width="2" />
</svg>
```



Obr. č. 14 - obrázek k danému kódu

5.5.3. Textury

Textury se vytvářejí pomocí elementu `<pattern>`, kdy se předdefinované grafické prvky opakují podél obou os tolikráte, dokud nezaplní celou definovanou plochu objektu.

Atributy elementu `pattern`:

1) `patternUnits` – vybere jeden ze dvou souřadnicových systémů pro atributy definující grafický vzorek (`userSpaceOnUse` nebo `objectBoundingBox`)

2) `patternContentUnits` – definuje souřadnicový systém pro obsah vzorku
3) `patternTransform` – dodatečné souřadnicové transformace aplikující se na texturu
4) `xlink:href` – odkaz na jinou texturu

Ukázka:

```
<svg width="320" height="160" viewBox="0 0 800 400"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
<defs><pattern id="textura" patternUnits="userSpaceOnUse" x="0"
       y="0" width="100" height="100" viewBox="0 0 10 10">
      <path d="M 1 1 L 9 1 L 5 9 z" fill="forestgreen"
            stroke="greenyellow" />
      </pattern></defs>
<circle  fill="url(#textura)" stroke="seagreen" stroke-width="7"
         stroke-dasharray="8,2" cx="400" cy="200" r="150" />
</svg>
```

Obr. č. 15 - obrázek k danému kódu

5.5.4. Transformace

Transformace grafického prvku docílíme přidáním atributu `transform` do objektu. Na libovolný objekt lze aplikovat více než jednu transformaci.

Hodnoty atributu `transform`:

- `matrix(a,b,c,d,e,f)` – obecná transformační matice
- `translate(tx,ty)` – posun
- `scale(sx,sy)` – změna měřítka

- `rotate(uhel [stredx stredy])` – otáčení o zadaný úhel okolo libovolného středového bodu
- `skewX(uhel)` – zkosení podél osy `x`
- `skewY(uhel)` – zkosení podél osy `y`

Ukázka:

```
<svg width="550px" height="300px" viewBox="-30 -30 300 300"
      version="1.1" xmlns="http://www.w3.org/2000/svg">
<g transform="translate(30,30)">
   <text font-size="15" fill="red">
         velikost textu(15)</text></g>
<g transform="scale(2)">
   <text font-size="15" fill="red">
         velikost textu(15) zvětšena 2x</text></g>
<g transform="rotate(90)">
   <text font-size="20" fill="green">
         rotace o 90°</text></g>
</svg>
```

Obr. č. 16 - obrázek k danému kódu

5.6. *Bitmapové efekty SVG*

Ve vektorové grafice se některé efekty dosahují velice obtížně, a proto zde přicházejí na řadu tzv. bitmapové efekty. Vektorová grafika se nejprve vyrastruje do pomocného paměťového bloku a až poté se aplikuje bitmapový efekt. V případě, že uživatel použije lupu na zvětšení grafiky dojde k novému výpočtu v daném rozlišení a nedojde ke ztrátě na kvalitě grafiky.

Pro vytvoření určitého filtru slouží element `<filter>`, který má své identifikační číslo a později se aplikuje na nějaký objekt.

Atributy elementu filter:

1) `filterUnits` - vybere jeden ze dvou systémů, pomocí kterých se bude definovat pracovní plocha filtru (hodnoty: `userSpaceOnUse` nebo `objectBoundingBox`)
2) `primitiveUnits` – definuje souřadnicový systém pro délkové údaje uvnitř filtru (`userSpaceOnUse` nebo `objectBoundingBox`)
3) `x, y, width, height` – určuje velikost pracovní plochy (výchozí hodnoty: `-10 %, -10 %, 120 %, 120 %`)
4) `filterRes` – určuje rozlišení bitmapy, ve které probíhá výpočet filtru
5) `xlink:href` – odkaz na jiný filtr

Aktivace filtru:

```
<filter id="nazev_filtru" filterUnits="hodnota" x="hodnota"
        y="hodnota" width="hodnota" height="hodnota">
<!—obsah filtru -->
</filter>

<rect filter="url(#nazev_filtru)"/>
```

Elementární grafické filtry (filter primitives):

1) Efekty osvětlení
   - `feDistantLight` – vzdálený zdroj světla
   - `fePointLight` – bodové světlo
   - `feSpotLight` - reflektor
2) Elementární bitmapové filtry
   - `feBlend` - prolínání dvou bitmap
   - `feComposite` - sloučení dvou bitmap
   - `feMerge` - sloučení dvou bitmap

- `feComponentTransfer` - úprava jasu
- `feColorMatrix` - úprava barevnosti
- `feConvolveMatrix` - kombinování sousedních bodů
- `feFlood` - vyplnění plochy
- `feImage` - vložení další grafiky
- `feOffset` - posuv
- `feDisplacementMap` - deformace
- `feTile` - vyplnění vzorkem
- `feGaussianBlur` - rozostření
- `feMorphology` – zesílení nebo zúžení
- `feTurbulence` - generování šumu
- `feSpecularLighting` - osvětlovací efekt
- `feDiffuseLighting` - osvětlovací efekt

Společné atributy všech filtrů:

1) `x, y, width, height` – nastaví konkrétní oblast pro použití filtru
2) `result` – pojmenovaný výsledek filtrové operce

3) `in` - určuje vstup filtru a nabývá hodnot:
   - `SourceGraphic` – vstupem je původní RGBA grafika
   - `SourceAlpha` – vstupem je pouze alfa kanál původní zdrojové grafiky
   - `BackgroundImage` - obraz pozadí
   - `BackgroundAlpha` - průhlednost pozadí
   - `FillPaint` - výplň zdrojové grafiky
   - `StrokePaint` - výplň obrysů zdrojové grafiky

- název obrazové proměnné – definuje se v atributu `result` v předcházejícím filtru

5.6.1. Filtr `feDistantLight`

Vzdálený zdroj světla jehož paprsky mají v každém bodu grafiky stejný směr.

Atributy filtru:

- `azimuth` - směrový úhel osvětlení v rovině XY.
- `elevation` - směrový úhel osvětlení v rovině YZ.

5.6.2. Filtr `fePointLight`

Blízký zdroj světla, jehož paprsky mají v každém bodu grafiky jiný směr.

Atributy filtru:

- `x, y, z` - umístění zdroje v 3D prostoru

5.6.3. Filtr `feSpotLight`

Jedná se o zdroj světla typu reflektor, kdy intenzita osvětlení klesá od maximálně osvětleného středu směrem k vnějšímu úhlu.

Atributy filtru:

- `x, y, z` - umístění zdroje v 3D prostoru
- `pointsAtX, pointsAtY, pointsAtZ` - poloha bodu, na který je reflektor zaměřen
- `specularExponent` - hodnota exponentu, určujícího zaostření reflektoru
- `limitingConeAngle` - úhel omezující osvětlenou oblast

5.6.4. Filtr `feBlend`

Pomocí filtru `feBlend` lze docílit sloučení dvou bitmap pomocí režimu prolínaní.

Atributy filtru:

- `mode` – režim prolínaní (hodnoty: `normal`, `multiply`, `screen`, `darken`, `lighten`)
- `in2` – druhá vstupní bitmapa (obdobně jako u atributu in)

5.6.5. Filtr `feComposite`

Tímto filtrem docílíme zkombinování dvou bitmap.

Atributy filtru:

- `operator` – hodnoty: `over`, `in`, `out`, `atop`, `xor`, `arithmetic`
- `k1`, `k2`, `k3`, `k4` - konstanty pro výpočet (u hodnoty `arithmetic`)
- `in2` - definuje druhou vstupní bitmapu

5.6.6. Filtr `feComponentTransfer`

Tento filtr se nabízí k použití pro operace typu úprava jasu, kontrastu a podobně.

Atributy filtru:

- `type` - typ funkce (hodnoty: `identity`, `table`, `discrete`, `linear`, `gamma`)
- `tableValues` - seznam hodnot
- `slope` - strmost přímky (výchozí hodnota – `1`)
- `intercept` - posun přímky na svislé ose funkce
- `amplitude` - amplituda gama funkce (výchozí hodnota – `1`)

- `exponent` - exponent gama funkce (výchozí hodnota – 1)
- `offset` - posun gama funkce

Ukázka:

```
<svg width="1500" viewBox="0 0 1900 440"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
<defs>
<linearGradient id="Gradient" gradientUnits="userSpaceOnUse"
      x1="300" y1="0" x2="1000" y2="0">
      <stop offset="0" stop-color="blue" />
      <stop offset="0.3" stop-color="black" />
      <stop offset="0.6" stop-color="pink" />
      <stop offset="1" stop-color="green" />
</linearGradient>
<filter id="Linear" filterUnits="objectBoundingBox"
        x="0%" y="0%" width="100%" height="100%">
        <feComponentTransfer>
        <feFuncR type="linear" slope="0.3" intercept="0.9"/>
        <feFuncG type="linear" slope="0.2" intercept="0"/>
        <feFuncB type="linear" slope="0.3" intercept="0.2"/>
        </feComponentTransfer>
</filter>
</defs>
<g font-family="Verdana" font-size="72" font-weight="bold"
      fill="url(#Gradient)">
      <text x="10" y="100" filter="url(#Linear)">
      Filtr feComponentTransfer s typem linear
</text>
</g>
</svg>
```

Obr. č. 17 - obrázek k danému kódu

5.6.7. Filtr `feConvolveMatrix`

Tímto filtrem docílíme například zostření nebo rozostření díky vzájemné kombinaci sousedících obrazových bodů.

Atributy filtru:

- `order` - rozměr konvoluční matice
- `kernelMatrix` - vlastní obsah konvoluční matice (kernelu) zapsán po řádcích
- `divisor` - dělitel je součtem všech členů matice
- `bias` - konstanta, která se přičte k výsledné hodnotě
- `targetX`, `targetY` - relativní X nebo Y poloha konvoluční matice vůči pixelu, pro který zrovna probíhá výpočet
- `edgeMode` - určuje způsob výpočtu na okrajích (hodnoty: `duplicate` - zopakují se okrajové body, `wrap` - vezmou se hodnoty z opačného okraje, `none` – použijí se nulové hodnoty)
- `kernelUnitLength` – nastavení velikosti jedné buňky matice
- `preserveAlpha` – `false` - aplikace filtru na všechny čtyři obrazové kanály, `true` - zachován kanál průhlednosti

5.6.8. Filtr `feOffset`

Tímto filtrem docílíte posunutí bitmapového rastru ve směru osy X nebo Y. Je vhodný k výrobě stínů.

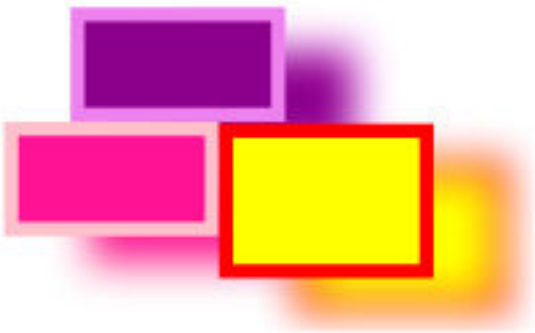
Atributy filtru:

- `dx` - posun ve směru osy X
- `dy` - posun ve směru osy Y

Ukázka:

```
<svg width="320px" viewBox="0 0 320 200"
  xmlns="http://www.w3.org/2000/svg" version="1.1"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<filter id="stin" x="-2%" y="-4%" width="128%" height="128%">
    <feOffset in="SourceGraphic" dx="40" dy="20" />
    <feGaussianBlur result="blur" stdDeviation="10" />
    <feBlend in="SourceGraphic" in2="blur" mode="normal" />
</filter>
<g stroke-width="7" filter="url(#stin)"
image-rendering="optimizeSpeed">
```

```
    <rect x="53" y="10" width="100" height="50"
      stroke="violet" fill="darkmagenta" />
    <rect x="20" y="67" width="100" height="50"
      stroke="pink" fill="deeppink" />
    <rect x="127" y="68" width="100" height="70"
      stroke="red" fill="yellow" />
</g>
</svg>
```



Obr. č. 18 - obrázek k danému kódu

5.6.9. Filtr `feTile`

Tento filtr lze použít pro vyplnění cílové plochy opakujícími se kopiemi zdrojové bitmapy.

Atributy filtru:

- `x, y, width, height` - vyplňovaná oblast

Ukázka:

```
<svg width="300px" viewBox="0 0 300 100" version="1.1"
     xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink" >
<defs>
<filter id="filtr" filterUnits="userSpaceOnUse">
     <feImage xlink:href="pic.gif" x="8" y="0" width="78"
      height="42" result="img"/>
     <feTile x="0" y="0" width="312" height="126" in="img"/>
</filter>
</defs>
<rect x="0" y="0" width="312" height="126" filter="url(#filtr)"/>
</svg>
```

Obr. č. 19 - obrázek k danému kódu

5.6.10. Filtr `feGaussianBlur`

Tento filtr je jeden z nejpoužívanějších efektů – Gausovské rozostření.

Atributy filtru:

- `stdDeviation` - velikost rozostření, uvedeny mohou být dvě hodnoty (pro každou osu zvlášť)

Ukázka:

```
<svg width="320px" viewBox="0 0 330 300"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns="http://www.w3.org/2000/svg"
     preserveAspectRatio="xMidYMid meet">
<g><defs>
   <g id="rects">
      <rect x="0" y="0" width="90" height="90" fill="violet"/>
      <rect x="45" y="45" width="90" height="90" fill="deeppink"/>
   </g>

<filter id="blur" filterUnits="objectBoundingBox" x="-25%"
            y="-25%" width="150%" height="150%">
<feGaussianBlur stdDeviation="21"/>
</filter>
</defs>
<use x="0" y="15" xlink:href="#rects"/>
     <g transform="translate(150,15)">
     <use xlink:href="#rects" filter="url(#blur)"/>
     </g>
</g>
</svg>
```

Obr. č. 20 - obrázek k danému kódu

5.6.11. Filtr `feSpecularLighting`

Tento filtr používá *Phongův* osvětlovací model k simulaci nasvícení trojrozměrného povrchu a počítá takzvanou *specular* složku (odlesky). Filtr může počítat osvětlení způsobené zdrojem, který je definován pomocí vnořeného prvku `feDistantLight`, `fePointLight` nebo `feSpotLight`.

Atributy filtru:

- `surfaceScale` - měřítko určující velikost nerovností osvětlovaného povrchu
- `specularConstant` - specular konstanta pro odraz světla
- `specularExponent` - specular exponent, čím vyšší hodnota, tím vyšší lesklost povrchu (rozsah od 1.0 do 128.0)
- `lighting-color` - definuje barvu světla
- `kernelUnitLength` – nastavení velikosti jedné buňka matice

Ukázka:

```
<svg width="320px" viewBox="0 0 250 200"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
<defs>
<filter id="point" x="0%" y="0%" width="100%" height="100%">
      <feGaussianBlur stdDeviation="1"/>
      <feSpecularLighting surfaceScale="3"
            lighting-color="hotpink" specularConstant="3"
            specularExponent="9">
      <fePointLight x="125" y="75" z="25"/>
      </feSpecularLighting>
</filter>
</defs>
```

```
<rect x="0" y="50" width="250" height="150"/>
  <g filter="url(#point)">
    <rect x="0" y="50" width="250" height="150" fill="none"/>
    <text x="125" y="200" font-size="70" fill="red"
          font-weight="bold" text-anchor="middle">
     SUN</text>
  </g>
</svg>
```

Obr. č. 21 - obrázek k danému kódu

5.6.12. Filtr `feDiffuseLighting`

Tento filtr simuluje reálné osvětlení trojrozměrného povrchu a počítá takzvanou difúzní složku (stínování) *Phongova* modelu. Filtr může počítat zdroj osvětlení obdobně jako filtr `feSpecularLighting`.

Atributy filtru:

- `surfaceScale` - měřítko určující velikost nerovností osvětlovaného povrchu
- `diffuseConstant` - difusní konstanta pro odraz světla
- `kernelUnitLength` - nastavení velikosti jedné buňky matice

Ukázka:

```
<svg width="320px" viewBox="0 0 250 200"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
<defs>
<filter id="point" x="0%" y="0%" width="100%" height="100%">
      <feGaussianBlur stdDeviation="1"/>
      <feDiffuseLighting surfaceScale="3" lighting-color="hotpink"
        diffuseConstant="1">
      <fePointLight x="125" y="75" z="25"/>
      </feDiffuseLighting>
</filter>
</defs>
```

```
<rect x="0" y="50" width="250" height="150"/>
  <g filter="url(#point)">
    <rect x="0" y="50" width="250" height="150" fill="none"/>
    <text x="125" y="200" font-size="64" fill="black"
          font-weight="bold" text-anchor="middle">
     SUN</text>
  </g>
</svg>
```

Obr. č. 22 - obrázek k danému kódu

5.7. Animace

Ve formátu SVG lze vytvářet animace čtyřmi způsoby. První a nejpoužívanější je vytvoření animace za pomocí animačních elementů. Druhý způsob spočívá ve využití DOM (Document Object Model), u kterého lze za pomoci skriptů měnit atributy objektů. Třetí způsob využívá jazyk SMIL (Synchronized Multimedia Integration Language) a poslední způsob je založen na integraci jazyka SMIL, SVG a libovolného dalšího formátu založeného na XML.

Základní elementy pro animaci:

1) `animate` – mění hodnoty atributů v čase
2) `set` – změna nečíselných atributů
3) `animateMotion` – pohybuje zvoleným prvkem podél animační křivky
4) `animateColor` – změna barvy
5) `animateTransform` – animuje transformační atributy
6) `mpath` – definuje dráhu pohybu objektu

Základní atributy animačních prvků:

1) `xlink:href` – odkazuje na cílový element, který bude animován
2) `attributeName` – určení atributu, který se bude v animaci měnit
3) `attributeType` – určení jmenného prostoru XML, ve kterém je definován cílový atribut (hodnoty: `CSS`, `XML`, `auto`)
4) `from` – počáteční hodnota animovaného atributu
5) `by` – změna hodnoty vůči počátečnímu stavu (pokud nastavíme hodnotu `by` už se nenastavuje hodnota `to` a naopak)
6) `to` – konečná hodnota animace
7) `fill` – nastaví, zda výsledek animace zůstane zachován (`freeze`) nebo bude odstraněn (`remove`)
8) `additive` – nastaví, zda aktuální animovaná hodnota nahradí klidovou hodnotu (`replace`) nebo se k ní bude přičítat (`sum`)
9) `accumulate` – hodnota `sum` - k aktuální hodnotě animace se přičte hodnota z konce předcházejícího cyklu
10) `dur` – trvání animace
11) `repeatCount` – počet opakování animace
12) `repeatDur` – určuje dobu, po kterou se má animace opakovat
13) `type` – typy transformací u prvku `animateTransform`, jeho hodnoty jsou:
    - `translate(x y)` - posunutí
    - `scale(x y)` – měřítko pro osu X a Y
    - `rotate(uhel středx středy)` - otáčení
    - `skewX(x)`, `skety(y)` – nastavuje úhel zkosení
14) `begin` – definuje začátek animace a nabývá hodnot:
    - `offset-value` – čas vyjádřený v sekundách (`0s` – čas načtení SVG dokumentu)

- `syncbase-value` – umožní synchronizaci s koncem (`end`) nebo začátkem (`start`) jiné animace a lze přidat i časový posun (např.: `begin="id_animace.start+3s"`)
- `event-value` - tento typ umožní reagovat na události (např.: `begin="id_animace.repeatEvent"`)
- `accessKey-value` – reakce na stisk určité klávesy (např.: `begin="id_animace.accessKey('p')"`)
- `wallclock-sync-value` - spuštění v přesně definovaný reálný čas
- `indefinite` - umožní spouštět danou animaci buď voláním funkce `beginElement()` nebo klepnutím na hyperlink zacílený na daný animační prvek

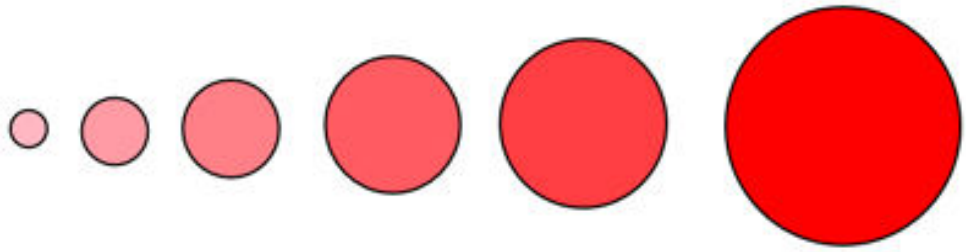
15) `end` – určuje čas ukončení animace (hodnoty stejné jako u `begin`)
16) `restart` – nastavuje restart animace (hodnoty: `never` – nikdy, `always` – restar je možný, `whenNotActive` – pokud animace není aktivní)
17) `calcMode` – definuje režim interpolace mezi jednotlivými vrcholy křivky pomocí hodnot:
    - `discrete` – skoková změna hodnoty mezi klíčovými body
    - `linear` – lineární přechod
    - `spline` - interpolace na základě Bézierových křivek (zadávají se klíčové řídicí body)
    - `paced` – konstantní rychlost po celou dobu trvání animace
18) `values` - seznam hodnot, kterých bude animace nabývat, jednotlivé kroky jsou odděleny středníky
19) `keyTimes` – ke každé hodnotě z atributu `values` lze nastavit určitý čas (hodnoty od 0 do 1)
20) `keySplines` – doplňující funkce k atributu `keyTimes`, jde o sady čtyř řídicích bodů `x1 y1 x2 y2` v rozsahu 0 až 1, které

pomocí Bézierových křivek určují dynamiku změny hodnot mezi dvěma klíčovými časy

21) `keyPoints` - určuje, jak daleko na cestě se má objekt nacházet v příslušném čase (uloženém v atributu `keyTimes`)
22) `path` - data pro vektorovou cestu určující pohyb objektu
23) `rotate` – způsob otáčení objektu ve vztahu k cestě
    - `auto` – objekt se natáčí ve směru tečny pohybové křivky
    - `auto-reverse` – objekt je otočen o 180° a natáčí se ve směru tečny pohybové křivky
    - `uhel` – [hel natočení

Ukázka:

```
<svg width="200" height="200" viewBox="0 0 200 200"
      xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
<circle cx="100" cy="100" r="50" stroke="black" stroke-width="2">
<animateColor attributeName="fill" attributeType="XML" begin="0s"
              dur="30s" fill="freeze" from="pink" to="red" />
<animate attributeName="r" attributeType="XML" begin="0s"
      dur="30s" fill="freeze" from="10" to="90" repeatCount="1" />
 </circle>
</svg>
```



Obr. č. 23 - obrázek k danému kódu

6. SVG v praxi

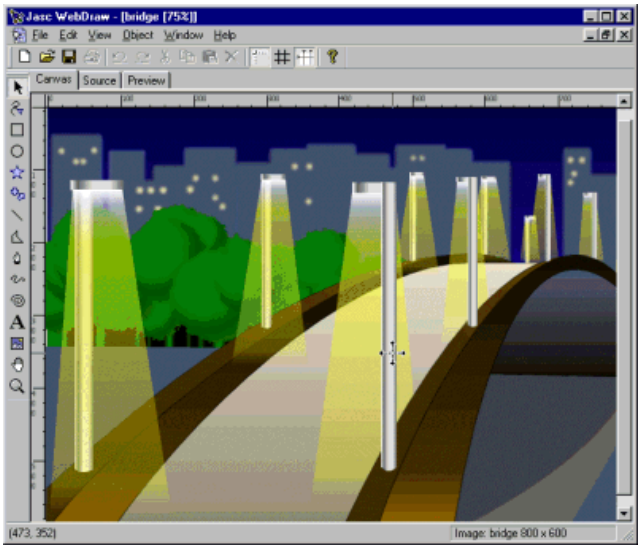
6.1. Editory

6.1.1. Editory s nativní podporou SVG

Editory s nativní podporou SVG mají v sobě přímo implementovanou podporu SVG a nepotřebují žádný zásuvný modul. Jejich základní výstupní formát je většinou SVG.

Editory s nativní podporou:

1) Inkscape 0.46 – open-sourcový grafický editor
2) Amaya 10.0 – open-sourcový editor konsorcia W3C
3) RapidSVG v2.0 Professional – komerční WYSIWYG editor
4) Jasc WebDraw – komerční editor
5) Sodipodi 0.34 – dobře funkční open-sourcový editor
6) XStudio
7) Ikivo Animator



Obr. č. 24 – Jasc WebDraw

6.1.2. Editory podporující SVG

1) Adobe Illustrator CS3
2) Corel Draw X3

6.2. Podpora SVG v prohlížečích

6.2.1. Internet Explorer

Internet Explorer 7 má částečnou podporu formátu SVG a pokud chceme zobrazit SVG grafiku v Internet Exploreru je nutné si nainstalovat plug-in SVG Viewer od firmy Adobe. Dále nepodporuje přímé vkládání SVG grafiky do XHTML kódu webové stránky a je proto nutné použít jmenný prostor `svg:` (např.: místo `<line />` se použije `<svg:line />`). Poté musíme SVG grafiku „svázat“ se jmenným prostorem, protože ani SVG Viewer jí nepozná, to zajistí následující kód vložený někam na začátek dokumentu:

```
<object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-
3cd390eeb4e2"></object>
<?import namespace="svg" implementation="#AdobeSVG"?>
```

6.2.2. Mozilla Firefox

U prohlížeče Mozilla Firefox se dá říci, že podpora je o něco lepší než u Internet Exploreru, ale není ovšem úplná. Vkládání SVG grafiky přímo do kódu webové stránky sice funguje, ale zato nefungují některé filtry a animace.

Přehled podporovaných elementů SVG ve Ferifoxu:

http://developer.mozilla.org/en/docs/SVG_in_Firefox

6.2.3. Opera

Nejlépe obstála v podpoře SVG Opera. Podporuje jak vkládání SVG grafiky přímo do XHML kódu webové stránky, tak filtry i animace.

Přehled podporovaných elementů SVG v Opeře:

http://www.opera.com/docs/specs/svg/

6.3. Vkládání SVG do stránek

SVG grafika může být vložena jako samostatný soubor nebo umístěna do XHTML/XML souboru.

6.3.1. Vkládání samostatného souboru

Pro první způsob použijeme element `<object>`. V případě, že software pro zobrazení není k dispozici můžeme nabídnout alternativní řešení jako rastrový obrázek nebo chybové hlášení.

Ukázka kódu:

```
<object width="256" height="256"
    data="logo.svg" type="image/svg+xml"
    standby="Please wait, loading from the Net!">
  <p>CHYBA!</p>
  <p>NELZE zobrazit SVG grafiku!</p>
  <h2>Prosím, nainstalujte na tento počítač
      <a href="http://www.adobe.com/svg/viewer/install/">
      SVG plug-in</a>!
  </h2>
</object>
```

Problém ovšem nastává u prohlížečů MS Internet Explorer, kdy Adobe plug-in v této konfiguraci při stahování grafiky z internetu nemůže nalézt žádný soubor připojený uvnitř SVG kódu. Proto je nutné použít prozatímní řešení pomocí elementu `<embed>`.

Ukázka kódu:

```
<embed src="logo.svg" type="image/svg+xml"
    pluginspage="http://www.adobe.com/svg/viewer/install/"
    height="128" width="128">
```

6.3.2. Vložení do XHTML/XML souboru

Druhý způsob vkládání SVG grafiky přímo do XHTML/XML souboru je poněkud složitější, protože u MS Internet Exploreru nastávají opět problémy.

MS Internet Explorer je schopen XHTML+SVG dokumenty zobrazit pouze tehdy, mají-li příponu *.htm/.html.* Zatímco ostatní prohlížeče vyžadují správnější přípony *.xml/.xhtml*.

V poslední řadě je velice důležité nastavit správné DTD dokumentu a to na XHTML + SVG.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus
SVG 1.1//EN"
"http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
```

6.3.3. Optimální řešení

Za nejoptimálnější řešení by se dalo považovat použití podmíněného komentáře, pomocí kterého nabídneme MS Internet Explorer zastaralý element *embed* a všem ostatním prohlížečům *object*.

```
<!--[if IE]>
...
<![endif]-->
<![if !IE]>
...
<![endif]>
```

7. Závěr

V mé absolventské práci jsem nejdříve popsala bitmapovou a rastrovou grafiku, abych blíže přiblížila výhody, nevýhody a základní rozdíly mezi nimi. Dále jsem popsala stručně jazyk XML, z kterého SVG vychází.

V hlavní částí práce jsem se zabývala základními informacemi o SVG a jeho historií. Dále následuje průvodce SVG. Podrobně jsem se věnovala struktuře dokumentu SVG, vytváření grafických primitiv, textu a jeho vlastnostem a v neposlední řadě jsem zmínila efekty v SVG a animaci.

V závěrečné části jsem se zaměřila na dostupné editory pro SVG a zhodnotila podporu v prohlížečích.

Práce na absolventské práci mě velice bavila, protože jsem získala plno nových informací a dozvěděla jsem se mnoho o technologii SVG.

8. Použité zdroje

[1] http://www.w3.org/

[2] http://www.svgx.org/

[3] http://www.interval.cz/

[4] http://www.root.cz/

[5] http://www.zive.cz/

9. Přílohy