Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Studijní obor: Výpočetní technika a informatika

Technologie SVG – aktuální standard webové vektorové grafiky

Tereza Skleničková

Vedoucí práce

PaedDr. Petr Pexa

Rok 2009

Zadání práce:

V bakalářské práci bude komplexně zpracována aktuální technologie SVG pro tvorbu webové vektorové grafiky z pohledu webmastera-profesionála a bude provedeno porovnání s obdobnými technologiemi pro tvorbu webové vektorové grafiky (např. Macromedia Flash). Cílem bude vytvořit první ucelenou publikaci v ČR, zabývající se touto problematikou, včetně vytvoření sady praktických příkladů formou webové prezentace.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V ....., dne .....

.....

podpis

Poděkování

Děkuji za konzultace a potřebné, nepostradatelné rady vedoucímu práce panu PaedDr. Petru Pexovi.

Anotace:

Cílem této práce je popsat technologii SVG a na konkrétních příkladech demonstrovat praktické využití webové vektorové grafiky.

Práce bude obsahovat informace o problémech se zobrazením v prohlížečích a nabídne přehled možných editorů, které můžeme pro vytváření webové vektorové grafiky používat. k praktickému využití této technologie je důležité znát sadu jednoduchých technik, návodů k vytváření křivek a znalost vlastností základních geometrických tvarů a různých efektů

Klíčová slova

Vektor, webová vektorová grafika, xml, bitmapová grafika, technologie SVG, historie a vývoj SVG, , základní grafická primitiva, geometrické tvary, vlastnosti geometrických tvarů, textury, efekty, křivky, editory, podpora svg v prohlížečích, skalární vektorová grafika

Abstract:

Purpose of this paper is to describe SVG technology and demonstrate its practical using by particular examples. Document will contain informations about problems with displaying in browsers and offers survey of available editors, which we can use for work with web vector graphics.For practical use of this technology is important to know a set of easy techniques, instructions for creation of curves and knowledge of properities of basic geometrical shapes and various effects.

Keywords

Vector, web vector graphics, xml, bitmap graphics, SVG Technologies, history and development SVG, basic grafic primitivs, geometric shapes , characteristics of geometric shapes, texture, efects, behaviours, editors, support svg in web browsers, Scalable Vector Graphics

1 Úvod:

SVG (z anglického *Scalable Vector Graphics* škálovatelná vektorová grafika) je značkovací jazyk a formát souboru, který popisuje dvojrozměrnou vektorovou grafiku pomocí XML. Formát SVG by se měl v budoucnu stát základním otevřeným formátem pro vektorovou grafiku na Internetu. Zatímco pro rastrovou grafiku je na Internetu formátů dostatek (např. GIF, PNG a JPEG), otevřený vektorový formát zatím na Internetu chyběl.

SVG definuje tři základní typy grafických objektů:

- vektorové tvary (vector graphic shapes – obdélník, kružnice, elipsa, úsečka, lomená čára, mnohoúhelník a křivka)
- rastrové obrazy (raster images)
- textové objekty

Tyto objekty mohou být různě seskupeny, formátovány pomocí atributů nebo stylů CSS a polohovány pomocí obecných prostorových transformací. SVG též podporuje ořezávání objektů, alpha masking, interaktivitu, filtrování obrazu (konvoluce, displacement mapping, atd…) a animaci. Ne všechny SVG prohlížeče však umí všechny tyto vlastnosti.

Cílem mé bakalářské práce je přiblížit rozdíly mezi bitmapovou a vektorovou grafikou, poukázat na výhody použití SVG souborů k tvorbě webové prezentace a v neposlední řadě přidám širokou galerii praktických příkladů využití SVG.

Nedílnou součástí práce je samozřejmě pohled do historie techniky SVG, nezapomenu na základní primitiva a jejich vlastnosti, různé efekty a animace. Ráda bych se také zabývala některými editory a v neposlední řadě podporou SVG v prohlížečích.

2 Teoretický úvod

SVG je jazyk, který popisuje dvojrozměrnou grafiku pomocí XML.. Předpokladem pro tvorbu SVG dokumentu je alespoň základní znalost **HTML** a **CSS** - zápis SVG je strukturou podobný právě HTML, ovšem se specifickými grafickými elementy. Jak HTML, tak SVG jsou vlastně v moderním pojetí speciálními případy **XML**.

2.1 SVG znamená…

SVG je zkratkou anglického výrazu Scalable Vector Graphics:

- **Scalable** - zdůrazňuje možnost libovolného zmenšování či zvětšování bez ztráty kvality a detailů obrazu. To znamená, že měřítko SVG obrazu může být kdykoli změněno podle potřeby, skript jej například může automaticky přizpůsobit velikosti obrazovky vašeho počítače, nebo třeba displeji mobilního telefonu či PDA. Stejný SVG dokument lze vytisknout bez jakýchkoli "zubů" na laserové tiskárně s vysokým rozlišením atd.
- **Vector Graphics** - vektorová grafika obsahuje matematicky definované geometrické objekty tvořené z linek a křivek, které nám umožňují tvořit SVG dokumenty v podstatě bez omezení.

2.2 Grafické schopnosti

SVG definuje tři základní typy grafických objektů.

- Vektorové tvary (vector graphic shapes),
- Rastrové obrazy (raster images)
- Textové objekty

Objekty mohou být formátovány atributy nebo styly CSS, seskupovány a polohovány pomocí obecných prostorových transformací.

Objekty uvnitř elementu SVG jsou vykreslovány ve stejném pořadí, ve kterém jsou zapsány ve zdrojovém textu, přičemž mohou mít přesně definovanou průhlednost, ořezovou cestu (clip mask) nebo bitmapovou masku (alpha masks). Významných úspor paměti lze dosáhnout použitím symbolů, kdy se jednou definovaný vzorový objekt použije mnohonásobně pomocí odkazů.

Na libovolný grafický element lze aplikovat velmi flexibilní bitmapové efekty. Přitom je takový element stále v původní vektorové podobě a teprve při vykreslování

v prohlížeči (on the client side) je například rozostřen. Můžete kupříkladu napsat filtr, který, při aplikaci na libovolný element, vytvoří prostorový stínovaný vzhled s vrženým měkkým stínem.

Text je text - textové informace uvnitř SVG grafiky zůstávají, stejně jako v HTML dokumentech, stále textem s možností fulltextového vyhledávání, nebo třeba kopírování vybraného textu do textového editoru. Na rozdíl od HTML lze ovšem velmi přesně zachovat tvar a formátování písma tak, že typografie bude stejná na libovolném počítači, kde si dokument budete prohlížet. SVG například exaktně definuje způsob, jak uložit tvary některých písmen (font subset) nebo všech znaků daného řezu písma a pak je použít pro zobrazení na počítači, kde originální písmo nemusí být dostupné. Používat lze samozřejmě také různé exotické způsoby psaní, jako zprava doleva a podobně. Podporováno je i univerzální kódování UNICODE.(1)

2.3 Animace a interaktivita

Máte možnost nadefinovat tzv. **deklarativní animace**. To znamená, že máte k dispozici pohyblivý grafický objekt, který nemusíte ošetřovat žádným programovým kódem (obdobně jako animovaný GIF v HTML). Tento způsob animace je kompatibilní se standardem **SMIL** (Synchronised Multimedia Integration Language). Deklarativní animace byly zavedeny kvůli požadavkům grafiků z praxe a kromě jiných výhod umožňují úpravy animací v různých editačních aplikacích.

Musím připomenout vlastnost, která souvisí přímo s XML základem SVG grafiky. Scripty na straně prohlížeče totiž mohou pomocí standardního rozhraní *(DOM)* naprosto libovolně měnit nebo dokonce vytvářet nové grafické objekty! Stejně fungují standardní mechanismy událostí (events). Jde tedy o podobnou techniku jako dynamické HTML.

3 Historie a vývoj

SVG patří mezi ty formáty založené na XML, které mají největší naději, že se stanou všeobecně akceptovaným standardem.

SVG (Scalable Vector Graphics), značkovací jazyk pro dvourozměrnou vektorovou grafiku, pochází z dílny W3C, které se stará o specifikaci standardů pro web. Po prvním návrhu specifikace, který byl zveřejněn v lednu roku 1999, následoval běžný schvalovací proces. Ten vyvrcholil uvolněním verze 1.0. Od 5. září 2001 tak SVG 1.0 konečně dostalo označení W3C Recommendation, a bylo oficiálně zpřístupněno pro všeobecné použití. Tim Berners-Lee, ředitel W3C a „zakladatel“ webu, při příležitosti uvolnění specifikace SVG uvedl, že „pomocí SVG se grafika na webu změní z pouhé dekorace na opravdovou grafickou informaci“.

SVG 1.0 je postaveno na základech dalších specifikací W3C, jako je DOM (Document Object Model), CSS, XSL, RDF, XML Linking nebo SMIL Animation. SMIL (Synchronized Multimedia Integration Language) je standard pro synchronizaci multimediálních procesů a bylo W3C doporučeno ve stejný den jako SVG. Při vývoji SVG se nezačínalo od úplného začátku. Byly využity už dříve vypracované návrhy specifikací pro vektorovou grafiku, jako je na PostScriptu založeném jazyk PGML od Adobe (Precision Graphics Markup Language) nebo VML (Vector Markup Language) podporovaný Microsoftem.

Představu, jakou má SVG perspektivu z hlediska zájmů komerční sféry, může ozřejmit seznam společností, které se pod hlavičkou W3C na vývoji SVG podílejí: Adobe Systems, AOL/Netscape, Apple, Autodesk, Bitflash, Canon, Corel, CSIRO, Eastman Kodak, Ericsson, Excosoft, Hewlett-Packard, IBM, ILOG, IntraNet Systems, KDDI, Macromedia, Microsoft, Nokia, OASIS, Openwave, Opera, Oxford Brookes University, Quark, Savage Software, Schemasoft, Sun Microsystems, Xerox a ZoomOn.

Pouhý seznam zvučných jmen ovšem není rozhodující. Skutečně důležitým činitelem pro prosazení SVG je jeho podpora v aplikacích. z hlediska masového rozšíření na webu je nejpodstatnější podpora v prohlížečích; zdaleka nejpoužívanějším prohlížečem je MS Internet Explorer. Dave Massy vysvětluje strategii Microsoftu: prohlížeč podporuje ty standardy, které jsou důležité pro zákazníka. Takže zatímco moderní standardy DOM Level 1 a CSS Level 1 nový prohlížeč plně podporuje, protože slouží všem uživatelům, ostatní jako SVG nebo

MathML ignoruje (otázkou je zda SVG, na rozdíl od MathML, není určen právě široké veřejnosti). Microsoft však umožňuje připojit k Exploreru zvnějšku nástroje ostatních výrobců, které s těmito standardy pracovat umí. To využila firma Adobe, která stojí v čele vývoje SVG a pro Internet Explorer vytvořila zásuvný modul (plug-in) Adobe SVG Viewer. Tento modul ve verzi 2.0 je možné volně stáhnout. Už teď je dispozici také beta verze Adobe SVG Viewer 3.0. Adobe včlenila SVG do svých hlavních produktů – např. obrázky nakreslené v Adobe Illustrator 9.0 je možné ukládat ve formátu SVG. Co se tedy týče podpory SVG Internet Explorerem, lze konstatovat, že Explorer prostřednictvím plug-inu od Adobe jazyku SVG rozumí. Problém rozšíření SVG se tak váže k otázce šíření Adobe SVG Vieweru mezi uživatele. (15)

3.1 SVG 1.1

Norma SVG 1.1 ze 14. ledna 2003 v podstatě pouze rozděluje komplexní "velkou" specifikaci SVG 1.0 na menší části v zájmu implementace SVG na méně výkonná (mobilní) zařízení. Definovány byly dva profily. **SVG Basic** je subset SVG normy určený pro PDA (omezeny jsou filtry a použití ořezových cest). Ještě „ořezanější“ **SVG Tiny** je určen pro mobilní telefony - vypouští scripty, filtry, přechody barev, vzorky, průhlednost a CSS formátování.

Povšimněte si jednoho nikoli nevýznamného detailu, v žádném případě nejsou omezeny animace! Je jisté, že je jim přikládána velká důležitost a jsou uživateli vyžadovány. SVG 1.1 má budoucnost nejenom v aplikacích pro mobilní komunikátory, ale může se uplatnit také pro nové implementace SVG, například nově vznikající nativní SVG funkce v internetových prohlížečích. Naprogramování minimálního subsetu SVG Tiny může být řádově snazší konstruktéři stránek, přitom budou velmi přesně vědět, jak bude vypadat konečné zobrazení.

Menší novinky SVG 1.1 zavádí v oblasti zpracování textu. Zlepšení již tak výborných typografických schopností představuje zavedení párového kerningu do definice SVG fontů CEF, dále podporuje ochranu autorství písem a umožňuje používání dalšího formátu internetových písem definovaného v normě CSS 2, tzv. WebFonts.(42)

3.2 Návrh SVG 1.2

Ještě v roce 2002 byl publikován první pracovní návrh SVG 1.2. Ten především přinášel nejvíce požadovanou vlastnost, kterou je schopnost automatického zarovnávání "tekoucího" textu. Funkce je navrhována podle vzoru v té době nejlepších profesionálních programů pro práci s textem (InDesign, QuarkXPress) tak, že by měla umět "natékat" text dokonce do celého řetězu textových rámů libovolného tvaru.

3.3 Podpora prohlížečů

Jak jste si všichni jistě všimli, podpora SVG v prohlížečích není zrovna valná a proto je ohledně SVG dokumentů stále spoustu otazníků, zda-li má tento formát budoucnost. Je známo pět nejpoužívanější prohlížečů a to Internet Explorer, Safari, Mozilla Firefox, Opera a Amaya. Začneme tedy postupně.

3.3.1 Internet Explorer vs. Safari

V článcích je uváděno, že podpora v Internet Explorer je od verze 5. Stejně jako v prohlížeči Safari byl pro správné zobrazování SVG dokumentů potřeba plugin, Adobe SVG Viewer, který nabízela firma Adobe a podporuje SVG 1.0/1. 1. Uživatelé Internet Explorer si musí tento plugin ručně stáhnout a nainstalovat. Plugin byl nabídnut i firmou Corel. Na rozdíl od IE prohlížeč Safari si potřebný plugin stáhne sám. Netrpělivě očekávaný Internet Explorer 8 sice přinesl řadu kýžených novinek ohledně podpory webových standardů a webových technologií. Podpora vektorového grafického formátu SVG mezi nimi ovšem nebyla. Všechny důležitější konkurenční prohlížeče přitom SVG podporují a právě jeho nepodpora ze strany stále ještě dominantního prohlížeče na trhu je udávána víceméně jako jediný důvod, který stojí v cestě k masovému rozšíření SVG.

Zato u nově připravované verze 9 je již více než pravděpodobné, že podpora SVG ze strany Exploreru konečně bude. Vzhledem k tomu, že verze 8, která vyšla v roce 2007 a do dnešního dne jsme se dočkali jen RC verze, budeme si muset asi na plnou podporu v tomto prohlížeči ještě pár let počkat.

3.3.2 Mozilla Firefox

Na rozdíl od IE je Mozilla o krok napřed. Pro Firefox 1.0 byl vytvořen návod jak nainstalovat SVG plugin. Pro uživatele je to nepohodlné na rozdíl od automatické instalace jako u Flashe. Funkce byla standardně vypnutá a pokud jste ji chtěli vyzkoušet, museli jste v about:config přidat položku svg.enabled a nastavit ji na true. Již od verze Mozilla Firefox 1.5 prohlížeč podporuje většinu SVG elementů.

Podporované elementy najdete na stránkách https://developer.mozilla.org/en/SVG_in_Firefox. . Je zřejmé že Mozilla Firefox nepodporuje kompletní SVG 1.1. k dispozici nejsou elementy k práci s fonty, většina filtrů a také chybí podpora animací. Tyto nedostatky by měly být podle tvůrců odstraněny, cílem je kompletní podpora SVG 1.1.

3.3.3 Opera

Přestože bylo SVG standardizováno roku 1998, jednotlivé prohlížeče většinou nepodporují kompletní specifikaci, ale pouze nějakou podmnožinu. Opera, která do své aplikace integrovala prohlížeč firmy Ikivo, podporuje od verze 8 pouze profil SVG 1.1 Tiny. v Opeře nelze pracovat s grafickými filtry, skripty, přechody barev, výplňovými vzorky, průhledností, symboly, ořezovými cestami, maskováním ani s některými složitějšími elementy pro zobrazení textu (<tspan>, <tref>). Opera, resp. SVG 1.1 Tiny nepodporuje ani formátování CSS. Autoři Opery předpokládají, že devátá verze bude obsahovat podporu standardu SVG 1.1 Basic. Podporované elementy najdete na stránce: http://www.opera.com/docs/specs/svg/ (30)

3.3.4 Amaya

Amaya je prohlížeč a editor webových stránek zároveň. Za jeho vývojem stojí organizace W3C, které zároveň stojí za vývojem většiny webových technologií včetně HTML, XHTML, CSS, SVG atd. Jeho největší výhodou a i důvodem, proč ho webdesignéři používají, je fakt, že generuje naprosto validní a čistý kód, který je přesně podle norem.

Zatím podporuje tyto technologie: HTML 4.01, XHTML 1.0, XHTML Basic, XHTML 1.1, HTTP 1.1, MathML 2.0, mnoho vlastností CSS 2 a SVG. **Amaya 10**

přináší podporu formátů SVG. Amaya 11.1 zejména opravuje chyby z předchozí 11 verze. Dále je například implementována částečná podpora SVG markers. (29)

3.4 Nejdůležitější editory

Pro grafiky a nadšence tu mám přehled některých z mnoha svg editorů.

3.4.1 Inkscape a Sketsa

Inkscape 0.44

Mezi vlastnosti, které je radno ocenit, patří maskování a ořezávání. k dispozici jsou také vrstvy nebo třeba matematické operace s vektorovými plochami (analogie cestáře z Illustratoru). Implementován je rovněž text na křivce.

Konečně máme také k dispozici pravé **SVG symboly**. Funkcí nazvanou "Clone" (v české verzi najdete pod "Úpravy/Klonovat/Vytvořit klon") vytvoříte libovolný počet kopií původního objektu a ve výstupním kódu vás každá kopie bude stát jeden kratičký příkaz use. Změníte-li originál, změní se přirozeně všechny jeho klony.

Inkscape běží na všech dnes používaných počítačových platformách.
Důležitým faktorem je také přátelské uživatelské prostředí. (35)

..

Sketsa 3.3

Stejně jako Inkscape pracuje tato javovská aplikace firmy Kiyut s SVG jako základním formátem pro ukládání grafiky v plné její šíři, to znamená, že zahrnuje i bitmapové efekty. Je určena hlavně pro programátory, takže představuje ideální nástroj pro "kodéry". Sketsa je zdarma k vyzkoušení, ale pro komerční využití je už nutno zaplatit licenci (shareware).

3.4.2 XStream RapidSVG a Ikivo Animator

XStream RapidSVG 1.0

Ze začátku nás může překvapit nezvyklé umístění některých funkcionalit. Například editace vektorových objektů na úrovni jednotlivých bodů není přístupná pomocí zvláštního nástroje, ale do tohoto editačního módu se dostanete po poklepání na dotyčný vektorový objekt. Polohové transformace objektů jsou zase dostupné pouze přes kontextovou nabídku. Kreslící funkce jsou uspořádány na nástrojové liště nazvané "Objects Bar". Dále máme k dispozici nezbytnou paletu "Library", kde

mohou být připraveny všechny často používané objekty a zvuky. Program není schopný vytvářet pravé SVG symboly, které mohou významně pomoci při optimalizaci velikosti grafických souborů. Velikou výhodou této aplikace je mimo animací rovněž interaktivita, ruku v ruce s vytvářením kompletních vícestránkových prezentací. SVG designérům je umožněno používat základní **formulářové prvky**, které SVG samo o sobě neobsahuje - například tlačítka, zaškrtávací políčka, textová pole a podobně. Funkčnost těchto formulářů zajišťuje Rapid SVG pomocí dodávané javascriptové knihovny.

Ikivo Animator 1.0

Na prvním místě je třeba uvést, že program Ikivo Animator 1.0 je vyvíjen především pro potřeby mobilních aplikací a podporuje pouze profil **SVG Tiny**.

Hlavní rysy:

1. animace - intuitivní vytváření nahráváním
2. bitmapové efekty (prvky filter) - NE
3. symboly (opakované použití objektů pomocí "use") - NE
4. složené cesty - ANO
5. text na křivce - NE
6. ořezávání - NE
7. maskování - NE
8. hyperlinky - ANO
9. interaktivita - NE
10. interaktivita pomocí připojení podprogramů JavaScript - NE
11. přímá editace kódu - NE

Nadstandardní rysy:

1. výborná práce s animacemi, především typu animateMotion
2. SVG profily mobilních telefonů (37)

3.4.3 XStudio 6.1

XS je grafický editor s průměrnou kreslicí částí, ovšem velice nadstandardní v oblasti manipulace se specifickými vlastnostmi SVG formátu. Jeho schopnosti rovněž silně přesahují do oblasti kódování a programování - obsahuje například vyspělý programátorský editor s validátorem XML kódu a dokonce i debugger.

Hlavní rysy:

1. animace - ANO
2. bitmapové efekty (prvky filter) - NE
3. podporuje symboly (opakované použití objektů pomocí "use") - NE
4. složené cesty - ANO
5. text na křivce - ANO
6. ořezávání - ANO
7. maskování - NE
8. hyperlinky - ANO
9. interaktivita - ANO
10. interaktivita pomocí připojení podprogramů JavaScript - ANO
11. přímá editace kódu - ANO

Nadstandardní rysy:

1. praktické ovládání
2. JavaScript debugger
3. SVG-XML validátor
4. CSS editor
5. práce s transformačními atributy(38)

3.4.4 SVGDeveloper 1.0

.

Pracovní prostředí představuje mnoho komfortních nástrojů pro webové vývojáře a kodéry. Mimo jiné dobrá práce se zdrojovým kódem, přímé úpravy SVG-XML atributů nebo modifikace, případně správa stylů CSS pomocí dialogů a paletek programu.

Hlavní výhody ve zkratce

- výborné animace s využitím techniky klíčových bodů na časové ose
- podpora CSS
- výborná práce se zdrojovým kódem, včetně "napovídání" při psaní
- konverze bitmapových obrazů do kódu "base64"
- podpora symbolů
- podpora SVG přechodů a vzorků
- ořezávání(39)

3.4.5 XFY Basic Edition 1.3

Za zmínku rozhodně stojí editor XFY Basic Edition firmy Justsystems, který umožňuje vizuálně vytvářet kombinované XML dokumenty (se smíšenými jmennými prostory). Jedná se o software napsaný v Javě. Program má skutečně velmi rozsáhlé schopnosti pro úpravy zejména XHTML, ale i SVG a MathML dat.

K dispozici jsou různé způsoby pohledu na vytvářený dokument jako například zdrojový kód nebo stromová struktura. (39)

3.5 Profesionální editory a animátory SVG

1. **Beatware Mobile Designer 2**: *http://www.beatware.com*
2. **XStream RapidSVG 2**: *http://www.xstreamsoftware.com*
3. **Ikivo Animator 1.1**: *http://www.ikivo.com*
4. **SVG Developer 1.0**: *http://www.perfectsvg.com*
5. **Adobe GoLive CS2**: vizuální editor SVG Tiny a SMIL: h*ttp://www.adobe.com* (36)

3.6 Vývojové nástroje

SVG je podtřída XML, takže samozřejmě můžete využít kterýkoli z editorů XML, ale i jakýkoli textový editor. Hodně moderních editorů HTML a obecně editorů zdrojových kódů pravděpodobně bude umět alespoň barevně odlišovat různé prvky (elementy, atributy, data) zdrojové struktury.

- **Adobe Illustrator 10** - exportuje a importuje všechny statické vlastnosti SVG 1.0. Výborná podpora filtrů a jejich WYSIWYG editování. Špatná optimalizace textových objektů. (Komerční, MAC OS a Windows.)

- **Adobe InDesign 2**- v podstatě identické exportní možnosti s AI10, bez SVG filtrů. (Komerční, MAC OS a Windows.)

- **Bernard Herzog Sketch 0.6.14** - exportuje a importuje statické SVG. (Freeware, Linux/Unix.)

- **Bitflash Brilliance** - nativní SVG Tiny a SVG Basic editor, tzn. SVG je jeho vnitřní a základní grafický formát. Podporuje animace. (Komerční, Windows.)

- **Corel Draw 11** - exportuje a importuje statické SVG 1.0. Vytváření hyperlinků. (Komerční, MAC OS a Windows.)

- **JASC WebDraw 1** - nativní SVG editor. Podpora filtrů a jejich WYSIWYG editování. Umí dokonce SVG deklarativní animace. Integrace s ostatními programy od JASC. (Komerční, Windows, Linux se připravuje.)

- **Mayura Draw 4.1** - export SVG. (Shareware, Windows.) (42)

4 Struktura SVG dokumentu

Zde můžete prostudovat základní strukturu SVG dokumentu.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
3  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4  <svg xmlns="http://www.w3.org/2000/svg" version="1.1"
5  xmlns:xlink="http://www.w3.org/1999/xlink">
6  <-- SVG content goes here -->
7  </svg>
```

Řádek 1 Toto je XML deklarace , která specifikuje XML dokument verze XML 1.0 .

Řádek 2-3 Toto je typ SML dokumentu , deklarující jazykovou mutaci stránky. SVG gramatika je definována SVG 1.0 DTD.

Řádek 4 Toto je kořen XML dokumentu. Tento element také definuje standard XML názvu pro tento dokument. SVG element má mnoho atributů efektů celkového SVG dokumentu. Pro ujištění můžeme nahlédnout do specifikace SVG 1.0 pro popis těchto atributů a mnoho dalšího.

Řádek 5 Tato řádka definuje xlink jméno, které je použito jako reference externích obrázků a elementů uvnitř SVG dokumentu.

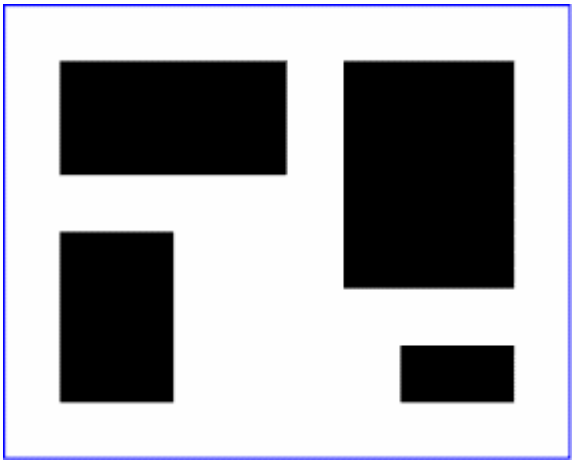
Řádek 6 Obsah SVG dokumentu bude zde.

Řádek 7 Tímto uzavíráme iniciaci svg elementů pro kompletní, správný a validní SVG dokument.

4.1 Základní element **svg**

Fragment (nebo také zlomek) SVG dokumentu definuje norma jako libovolný počet grafických prvků (elementů) obsažených v prvku svg. i když se mluví o zlomku, obsahuje všechny náležitosti potřebné ke korektnímu zobrazení. Fragment SVG dokumentu může existovat jako oddělený samostatný soubor (viz následující příklad), nebo může být zabudován uvnitř jiného rodičovského XML dokumentu (embedded inline).

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="5cm" height="4cm" version="1.1"
     xmlns="http://www.w3.org/2000/svg">
  <title>Samostatny SVG soubor</title>
  <rect x="0.5cm" y="0.5cm" width="2cm" height="1cm"/>
  <rect x="0.5cm" y="2cm" width="1cm" height="1.5cm"/>
  <rect x="3cm" y="0.5cm" width="1.5cm" height="2cm"/>
  <rect x="3.5cm" y="3cm" width="1cm" height="0.5cm"/>
  <!-- obrys kreslici plochy -->
  <rect x=".01cm" y=".01cm" width="4.98cm" height="3.98cm"
        fill="none" stroke="blue" stroke-width=".02cm" />
</svg>
```



Elementy svg se mohou objevit i uvnitř jiných SVG fragmentů, jinými slovy, je možné jejich vícenásobné vnořování. Lze toho využít například k nastavení vlastního souřadnicového systému pro úsek vnořeného kódu.

Jednotky: Povšimněte si prosím, že můžete používat v podstatě libovolné jednotky - zcela stejně, jako v CSS, tedy px, mm, cm, pt, em...

4.2 Atributy:

- ***x, y*** (souřadnice) - tyto atributy znamenají přesně to, co vyjadřuje jejich název (počátek souřadnic v levém horním rohu

- ***width*** (délka) - šířka kreslicího plátna; pokud není definována, použije prohlížeč hodnotu 100 % (výchozí nebo též default hodnota)

- ***height*** (délka) - výška kreslicího plátna; pokud není definována, použije prohlížeč hodnotu 100 % (výchozí nebo též default hodnota)

- ***viewBox*** (x y šířka výška) - atributy umožňují změnit souřadnicový systém uvnitř SVG fragmentu takto: x, y nastaví počátek virtuálních souřadnic (vlevo nahoře) uvnitř SVG fragmentu, šířka a výška pak určují rozsah virtuálních souřadnic (ideální například pro "napasování" grafiky do obdélníku o rozměrech, které předem nejsou známé - výborný prostředek k vytváření grafiky přizpůsobující se velikosti okna prohlížeče!); takový virtuální souřadnicový systém lze vytvořit též uvnitř elementů symbol a image

- ***preserveAspectRatio*** (typ_přizpůsobení_rozměrů) - použitím "viewBox" může dojít i k neproporcionální deformaci, proto máte k dispozici tento parametr, kterým můžete nastavit chování prohlížeče: "none" umožní deformaci souřadnic a hodnota, například "xMidYMid", přizpůsobí grafiku proporcionálně, tak, aby byla vidět celá, a navíc ji vycentruje v obou směrech (toto je výchozí a ve většině případů také nejvhodnější hodnota)

4.3 Seskupení – prvek **g**

Element g může sloužit jako schránka (container) pro skupinu vnořených objektů. Silně se doporučuje jeho použití ve spojení s elementy desc a title, k vytvoření strukturovaného a dobře zdokumentovaného grafického dokumentu. Což, mimo jiné, umožní zpřístupnit takové dokumenty i handicapovaným osobám (specializovaný prohlížeč by například mohl navigovat "hlasem"), ale může se velmi dobře uplatnit třeba i při popisu geografických objektů na SVG mapě a podobně.

Pomocí důvěrně známého atributu id se na grafické elementy můžete odkazovat, například pro potřeby deklarativních animací nebo skriptování.

4.4 Knihovní sekce – prvek **defs**

Defs je kontejnerový prvek pro elementy, na které se budeme později odkazovat. Jeho použití není striktně vyžadováno, ale důrazně doporučeno. Jeho chování a funkce je obdobná prvku g.

4.5 Knihovní objekt – prvek **symbol**

Prvek symbol definuje elementy pro vícenásobné použití (též knihovní prvky nebo prostě symboly), které jsou jednoznačně identifikovány dobře známým atributem id="jmeno_symbolu". Předdefinovaný prvek lze později mnohonásobně vkládat do kresby pomocí elementu use.

Ze strukturního hlediska je symbol podobný objektu g s následujícími zásadními rozdíly:

- sám o sobě se nikdy nevykreslí, pouze jeho instance (odkazy s použitím prvku use)
- má atributy viewBox a preserveAspectRatio
- lze používat v různých velikostech pomocí atributů (x, y, width, height) v prvku use

4.6 Použití objektu z knihovny – **use**

Element use odkazuje na knihovní prvek (symbol), který se zobrazí na místě grafické struktury, kde byl prvek use použit. Vznikne jakási virtuální kopie příslušného prvku symbol, která se ale **neobjeví** ve struktuře SVG DOM, tam zůstává pouze samotný objekt use!

Atributy:

- **xlink:href** - URI reference na symbol (knihovní objekt)
- **x, y** - poloha
- **width, height** - šířka a výška; nepovinné atributy, použity ke změně velikosti vkládaného symbolu

Pozor, animujete-li knihovní objekt (symbol), všechny jeho instance by při správné implementaci měly zůstat animované!

4.7 Obrázek – element **images**

Prvek images slouží pro vložení obrázku. Podle SVG normy můžete do grafického obsahu vložit nejen PNG, JPEG ale i další SVG soubor (v tomto případě jsou animace ignorovány).

V normě není zmíněn formát GIF, ale jisté je, že minimálně ASV (Adobe SVG Viewer) jej ve statické podobě podporuje včetně průhlednosti.

Atributy:

- **x, y** (souřadnice) - umístění
- width (délka) - šířka
- **height** (délka) - výška
- **preserveAspectRatio** (typ_přizpůsobení_rozměrů) - tímto parametrem lze nastavit jeho rozměrové přizpůsobení:

Při vkládání SVG se stejnojmenné parametry externího svg elementu ignorují a ty v elementu image dostávají přednost.

Výsledkem načtení a zpracování obrázku je vždy barevný model RGBA (čtvrtý kanál pro průhlednost), který je standardním interním modelem v SVG.

4.8 Zobrazovací a vykreslovací model

SVG používá takzvaný "malířův algoritmus" (**painters model**). Představu o počítačovém provedení si lze vytvořit snadno, stačí si uvědomit, že prvky z fragmentu SVG se vykreslují v přirozeném pořadí, tak jak jsou v XML dokumentu za sebou (to znamená, že první objekt bude opticky vespod, v pozadí - pozdější prvky jej překrývají).

Všichni už víme, že SVG zdaleka nejsou jen vektorové tvary v kombinaci s obrázky, jeho možnosti jdou ještě o třídu dál. Aby mohly prohlížeče správně vytvořit všechny speciální efekty, měly by aktuální element (případně celou skupinu elementů) vykreslit nejdříve v pomocném bloku paměti.

Nezapomínejme, ze se stále pracuje v modelu RGBA - tedy RGB barvy plus takzvaný "alfa kanál" pro uložení průhlednosti každého bodu. To je nutné nejen kvůli neomezeným možnostem maskování v normě SVG, ale i pro implementaci efektů (**SVG filters**). Filtry jsou totiž aplikovány právě na tuto pomocnou paměť. Obrovskou pružnost a kreativitu znásobuje fakt, že filtry můžete aplikovat nejen na kresbu, ale třeba i na kanál s průhledností.

Norma SVG explicitně nařizuje prohlížečům použití pomocné obrazové paměti pro prvek g. (Obsah bufferu je přitom inicializován na průhlednou černou barvu.) Až po kompletním vykreslení všech objektů ve skupině se na toto pomocné "plátno" aplikují různé efekty, a to přesně v následujícím pořadí:

1. bitmapové filtry
2. masky
3. průhlednost

Prohlížeče mají ztíženou úlohu tím, že g může být vnořeno mnohonásobně. Zdůrazňuji tento důležitý moment, jelikož jde o základní princip v SVG a je nutné, abyste jej měli vždy na paměti.

4. 9 Pravidla vykreslování objektu

Nejdříve se vykresluje výplň, poté tah a nakonec případné značky

Každá výplň a tah může přitom mít libovolnou na sobě nezávislou průhlednost.

Podporovány jsou následující typy vyplňování použitelné pro výplně i tahy:

- plné barvy
- přechody barev lineární a kruhové
- vzorky (textury) (2)

5 Formátování textu

5.1 KÓDOVÁNÍ TEXTU

Pro jiné než anglické texty vřele doporučuji kódování UTF-8, které by měly umět všechny SVG prohlížeče a neměli byste mít problémy ani s editováním takových souborů. Výborně je zvládá například pod MacOS editor BBEdit6+ nebo Linuxový KWrite z KDE3+.

5.2 ELEMENT TEXT

Základním prvkem pro vkládání textu je element `text`. Pro formátování se používají atributy z normy CSS a také mnoho dalších, které zatím pomineme.

```
<text x="250" y="150"
      font-family="Helvetica" font-size="55"
      fill="blue" >
  pokus
</text>
```

5.3 FONT a VELIKOST

Základní fonty jsou stejně jako v css 'serif', 'sans-serif', 'cursive', 'fantasy' a 'monospace'. Také můžete použít fonty s názvy "Times", "Baskerville", "Verdena", a "Symbol." . Zkoušela jsem různé názvy, ale nejspolehlivěji pracují ty základní. (41)

```
<svg width="15cm" height="10cm" viewBox="0 0 1000 300"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<desc> Text example
</desc>
<g fill = "red">
<text x = "10" y = "25" font-size = "30">
<tspan x = "10" dy = "20" font-family = "serif"> Toto je testovaci
text fontu serif.
</tspan>
<tspan x = "10" dy = "30" font-family = "sans-serif"> Toto je
testovaci text fontu sans-serif.
</tspan>
<tspan x = "10" dy = "30" font-family = "cursive"> Toto je
testovaci text fontu cursive.
</tspan>
<tspan x = "10" dy = "30" font-family = "fantasy"> Toto je
testovaci text fontu fantasy.
</tspan>
<tspan x = "10" dy = "30" font-family = "monospace"> Toto je
testovaci text fontu monospace.
</tspan>
<tspan x = "10" dy = "30"> Toto je testovaci text bez fontu.
</tspan>
</text>
</g>
</svg>
```

6 Základní grafická primitiva

6.1 OBDÉLNÍK

podívejte se na řádku s deklarací obdélníku

```
<rect x="100" y="100" width="400" height="200" fill="yellow"
stroke="black" stroke-width="3" />
```

Zde je popis jednotlivých vlastností:

- **rect** - obdélník je deklarován pomocí elementu rect.
- **x, y** - umístění levého horního rohu.
- **width, height** - šířka a výška obdélníku. v některých případech možnost vyjádření v procentech.
- **fill** - barva výplně.
- **stroke** - barva obrysu.
- **stroke-width** - tloušťka obrysu.

6.2 KRUH

Dalším tvarem je kruh, jeho deklarace je podobná jako obdélník.

```
<circle cx="100" cy="100" r="80" fill="orange" stroke="navy"
stroke-width="10" />
```

Zde je popis jednotlivých vlastností:

- **circle** - kruh je deklarován pomocí elementu circle.
- **cx, cy** - umístění středu.
- **r** - poloměr
- **fill** -barva výplně.
- **stroke** - barva obrysu.
- **stroke-width** - tloušťka obrysu.

6.3 ELIPSA

```
<ellipse transform="translate(300 200) rotate(-30)" rx="250"
ry="100" fill="none" stroke="blue" stroke-width="20"/>
```

Zde je popis jednotlivých vlastností:

- **ellipse** - elipsa je deklarován pomocí elementu ellipse.
- **cx, cz** - umístění středu.
- **rx, ry** - poloměry

6.4 ČÁRA

Jednoduchou čáru definují dva páry souřadnic "x", "y". Tento prvek také zcela logicky nemá výplň.

```
<line x1="100" y1="300" x2="300" y2="100" stroke-width="5"/>
```

Zde je popis jednotlivých vlastností:

- **line** - čára je deklarován pomocí elementu line.
- **x, y** - souřadnice.

6.5 LOMENÁ ČÁRA

Lomená čára je sada spojených jednoduchých linek (s otevřeným obrysem), jejichž parametry se zapisují do jediného atributu "points". Hodnoty mohou být odděleny mezerami nebo čárkami. Použita mohou být reálná čísla s desetinnou tečkou a exponentem (1.25e3) stejně jako u všech ostatních souřadnic v SVG. (3)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<desc>Zvetsujici se zuby</desc>
<polyline fill="none" stroke="blue" stroke-width="10"
points="50,375
150,375 150,325 250,325 250,375
350,375 350,250 450,250 450,375
550,375 550,175 650,175 650,375
750,375 750,100 850,100 850,375
950,375 950,25 1050,25 1050,375
1150,375" />
<!-- obrys pracovni plochy -->
<rect x="1" y="1" width="1198" height="398" fill="none"
stroke="blue" stroke-width="2" />
</svg>
```

Zde je popis jednotlivých vlastností:

- **polyline** - lomená čára je deklarován pomocí elementu polyline.
- **points** - parametry jednotlivých linek.

6.6 MNOHOÚHELNÍK

Mnohoúhelník - uzavřený vektorový tvar s libovolným počtem bodů, které se zapisují do atributu "points", jako u prvku polyline. (3)

```
<polygon fill="red" stroke="blue" stroke-width="10"
points="350,75 379,161 469,161 397,215
423,301 350,250 277,301 303,215
231,161 321,161" />
<polygon fill="lime" stroke="blue" stroke-width="10"
points="850,75 958,137.5 958,262.5
850,325 742,262.6 742,137.5" />
```

Zde je popis jednotlivých vlastností:

- **polygon** - mnohoúhelník je deklarován pomocí elementu polygon.
- **points** - parametry jednotlivých linek.

6.7 ELIPTICKÁ VÝSEČ

Vykreslí elipsoidní výseč ze současného bodu do bodu (x,y). Máme-li dány dva body, jako v našem případě, je možné jejich spojení po elipse čtyřmi způsoby. Parametr (vetsi-vysec) definuje, zda zvolí kratší (hodnota 0) nebo delší cesta (hodnota 1), zatímco "orientace-vysece" určuje směr „vypouklosti“ výseče .(3)

Zde je popis jednotlivých vlastností:

- **x,y**- nový bod výseče
- **rx,ry**- poloměry elipsy
- **x-rotace**- pootočení

```
<desc>Kolacovy graf s& 2 vysecemi a linka s& elipsovitymi
pulsy</desc>
<path d="M300,200 h-150 a150,150 0 1,0 150,-150 z" fill="red"
stroke="blue" stroke-width="5" />
<path d="M275,175 v-150 a150,150 0 0,0 -150,150 z" fill="yellow"
stroke="blue" stroke-width="5" />
<path d="M600,350 l 50,-25
a25,25 -30 0,1 50,-25 l 50,-25
a25,50 -30 0,1 50,-25 l 50,-25
a25,75 -30 0,1 50,-25 l 50,-25
a25,100 -30 0,1 50,-25 l 50,-25" fill="none" stroke="red" stroke-
width="5" />
<!-- obrys platna -->
<rect x="1" y="1" width="1198" height="398" fill="none"
stroke="blue" stroke-width="1" />
</svg>
```

6.8 CESTY

Pro obecné kreslení jsou k dispozici cesty neboli tahy.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 500 400" xmlns="http://www.w3.org/2000/svg"
version="1.1">
<title>Cubic Bézier
</title>
<desc>Jednoduchy priklad s& prikazy "C" a "S", popisky ridicich
bodu, k formatovani pouzity CSS styly
</desc>
<style type="text/css">
<![CDATA[ .Border { fill:none; stroke:blue; stroke-width:1 }
.Connect { fill:none; stroke:#888888; stroke-width:2 } .SamplePath
{ fill:none; stroke:red; stroke-width:5 } .EndPoint { fill:none;
stroke:#888888; stroke-width:2 } .CtlPoint { fill:#888888;
stroke:none } .AutoCtlPoint { fill:none; stroke:blue; stroke-
width:4 } .Label { font-size:22; font-family:Verdana } ]]>
</style>
<polyline class="Connect" points="100,200 100,100" />
<polyline class="Connect" points="250,100 250,200" />
<polyline class="Connect" points="250,200 250,300" />
<polyline class="Connect" points="400,300 400,200" />
<path class="SamplePath" d="M100,200 C100,100 250,100 250,200
S400,300 400,200" />
<circle class="EndPoint" cx="100" cy="200" r="10" />
<circle class="EndPoint" cx="250" cy="200" r="10" />
<circle class="EndPoint" cx="400" cy="200" r="10" />
<circle class="CtlPoint" cx="100" cy="100" r="10" />
<circle class="CtlPoint" cx="250" cy="100" r="10" />
<circle class="CtlPoint" cx="400" cy="300" r="10" />
<circle class="AutoCtlPoint" cx="250" cy="300" r="9" />
<text class="Label" x="25" y="70"> M100,200 C100,100 250,100
250,200
</text>
<text class="Label" x="325" y="350" style="text-anchor:middle">
S400,300 400,200
</text>
<!-- obrys platna -->
<rect class="Border" x="1" y="1" width="498" height="398" />
</svg>
```

Zde je popis jednotlivých vlastností:

- **m** - moveto (nastav aktuální polohu - bez kreslení čáry)
- **l** - lineto (kresli úsečku)
- **c** - curveto (kresli kubickou Bézierovu křivku)
- **q** - quadratic Bézier (nebudeme probírat)
- **a** - arc (eliptická výseč)
- **z** - closepath (uzavře aktuální vektorovou plochu)

Tyto příkazy se zapisují do atributu "d" (jako data) vždy jedním písmenem následovaným potřebnými souřadnicemi. (Pro zvýraznění budou v následujícím textu požadované souřadnice vždy uzavřeny v závorkách.) Oddělování příkazů i hodnot v rámci tohoto atributu je opět možné buď mezerami nebo čárkami.

Pokud cesta obsahuje více oddělených tahů (sub-paths), vzniká takzvaná složená cesta (compound path) - takto lze vytvořit například průhledný otvor uvnitř spojité plochy.

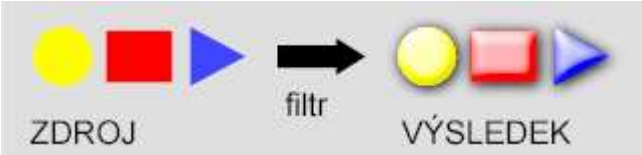
V dalším textu si povšimněte, že jednotlivé příkazy pro kreslení cesty mohou být zapsány jako malá (minusky) či jako VELKÁ písmena (verzálky). Tím se určí druh souřadnic - relativní či absolutní. (3)

7 Efekty

Přestože má vektorová grafika mnoho výhod, je zde stále jedno omezení. To spočívá v tom, že se pohybujeme pouze ve 2D prostoru. Pokud tedy chceme dosáhnout krásných, na první dojen působících jako 3D grafika, objektů, můžeme využít efektů,které SVG nabízí velkou škálu.

7.1 Bitmapové efekty SVG

Bitmapové efekty normy SVG si lze představit jako jakési dodatečné, následné zpracování (anglicky post-processing) vektorové grafiky - vektory se nejprve vyrastrují do nějakého pomocného paměťového bloku a na ten se poté aplikují dané efekty (filtry).



V prohlížeči může uživatel použít lupu - v této situaci dojde k novému výpočtu v aktuálním rozlišení, takže na "zubatou" grafiku zapomeňte.

Definice SVG dává grafikům do rukou potřebné nástroje k vytvoření v podstatě jakéhokoli efektu - počínaje takzvanými měkkými stíny, přes pohybové rozostření a šum, až po vytváření různých textur, plastické nápisy a podobně.

SVG sice má všechny potřebné nástroje v podobě elementárních bitmapových filtrů, ovšem většina těch nejzajímavějších efektů bude vyžadovat zkombinování více operací, přičemž se neobejdete bez poměrně pokročilých znalostí počítačové grafiky.

Při čtení dalšího textu budou proto mít výhodu ti, kdož si dobře rozumí s bitmapovým editorem Photoshop. v praktické části(CD)mám připravenu galerii některých zajímavých efektů. Hotové a odladěné filtry můžete ihned začít používat ve svých dokumentech. Mimochodem - předpřipravenou knihovnu SVG efektů obsahuje Adobe Illustrator, Mobile Designer i další kreslicí programy.

7.2 Podporované SVG prohlížeče

Filtry patří mezi pokročilé vlastnosti standardu **SVG Full** a jejich vykreslení zatím zvládají pouze prohlížeče ASV (ASV v Mozille a Firefoxu ve Windows) a Batik. Částečně podporuje filtry i nejnovější Corel SVG Viewer a probíhá také příprava nativní implementace v prohlížečích Mozilla, Firefox, Konqueror a Mac Safari {KSVG2}. Vývoj Mozilla SVG však bohužel stále pokulhává a zatím se zdá, že bychom si měli vsadit spíše na KSVG.

7.3 Teorie

Mějte stále na paměti, že se vždy pracuje v barvovém modelu RGBA - tedy tři základní RGB barvy plus takzvaný *alfa kanál* pro uložení průhlednosti každého bodu.

Norma SVG výslovně nařizuje prohlížečům použití pomocné obrazové paměti pro prvek `g` (seskupení objektů). Filtr můžete aplikovat i na jeden konkrétní prvek - v takovém případě SVG prohlížeč rovněž pracuje s pomocnou pamětí. (Obsah bufferu je přitom inicializován na průhlednou černou barvu.) Až po kompletním vykreslení všech objektů ve skupině se na tuto pomocnou paměť aplikují další operace v tomto pořadí:

1. **bitmapové filtry**
2. **masky**
3. **průhlednost**

Nikoli nevýznamným plusem takto definovaných efektů je zachování sémantické struktury grafických dokumentů, což je v moderním webdesignu věc velmi podstatná. Bezplatným bonusem navíc je možnost animování téměř všech parametrů filtrů pomocí takzvaných deklarativních animačních prvků SVG-SMIL! (23)

7.4 Objekt "filter"

Tento prvek slouží jako jakýsi kontejner, obsahující vlastní pracovní mechanismus SVG filtru. Jeho nejdůležitější funkcí je pojmenování výsledného efektu a rovněž nastavení rozlišení bitmapy, ve které budou probíhat výpočty elementárních filtrů.

Použiji-li analogii s Photoshopem, pak lze prvek filter přirovnat k vytvoření nového, prázdného obrazu.

```
<filter id='jmeno' filterUnits='objectBoundingBox'
  x='0%' y='0%' width='100%' height='100%'>
  <!-- obsah filtru zde -->
  ...
</filter>
```

Zatímco u vektorů pojem rozlišení v podstatě nemá smysl, pro bitmapové (alias rastrové) obrazy je rozlišení naopak stěžejním parametrem. Velmi zjednodušeně řečeno - čím vyšší rozlišení, tím více grafických detailů.

Zvyšujeme-li rozlišení bitmapové grafiky, roste spotřeba operační paměti s rozlišením kvadraticky. To samé platí o výpočetní náročnosti filtru.

V praxi je proto třeba velmi dobře uvážit tento parametr a nenastavovat rozlišení filtrů ani o pixel výše, než je nutné.

Atributy:

- ***filterUnits*** - nepovinný atribut určuje jednotky, pomocí kterých se bude definovat pracovní plocha filtru; možné hodnoty: userSpaceOnUse (výchozí hodnota, která přikazuje použít jednotky souřadnicového systému ve kterém je prvek filter vyvolán), objectBoundingBox (jednotky relativní k ohraničovacímu rámečku filtrované grafické skupiny, "100 %" odpovídá celkové velikosti skupiny).
- ***primitiveUnits*** - specifikuje souřadnicový systém pro délkové údaje uvnitř filtru; možné hodnoty: stejné jako u předcházejícího atributu.
- ***x, y, width, height*** - určuje velikost pracovní plochy filtru (filter effects region); výchozí hodnoty: -10 %, -10 %, 120 %, 120 %. Upozorňuji, že podivné výchozí hodnoty nejsou náhodné - je to proto, že výsledek některých operací - typicky rozostření - má větší rozměry než původní grafika. Častou chybou SVG grafiky, kterou lze nalézt na internetu , je právě *málo rozšířená plocha* pro

bitmapové výpočty, způsobující na hranách filtrovaných grafických prvků nehezké zubaté ukončení...

- ***filterRes*** - nepovinný atribut určuje rozlišení bitmapy ve které probíhá výpočet filtru, možno udat pro X a Y osu zvlášť.
- ***xlink:href*** - případný URI odkaz na další filtr (umístěný ovšem ve stejném SVG fragmentu). Jinak řečeno - lze libovolně vnořovat a kombinovat více oddělených prvků filter.

7.5 Používání efektů - atribut "filter"

Aktivace efektů SVG probíhá jejich zápisem do atributu filter, který lze přiřadit stejně tak ke skupině, jako třeba jen k jednomu konkrétnímu grafickému prvku.

Obsahem tohoto atributu je URI reference na prvek filter, který chceme aplikovat na příslušný grafický element nebo skupinu.

```
<rect ... fill="..." filter="url(#filtr)"/>

<g filter="url(#zase_filtr)"/>
...
</g>
```

7.6 Přístup k obrazu pozadí

Jedním z možných vstupů pro bitmapové efekty jsou dva speciální objekty **BackgroundImage** (pozadí RGBA) a **BackgroundAlpha** (pouze průhlednost pozadí), které představují původní obsah plochy, na kterou se náš filtr aplikuje.

Zavedení této vlastnosti má zvláštní dopad na implementaci prohlížeče, potažmo na systémové hardwarové zdroje. Je totiž nutné vytvořit kopii původní vykreslené oblasti a vyčlenit pro ni dodatečnou paměť.

Protože některá zařízení - obzvláště mobilní - disponují omezeným výkonem, musí tvůrce grafiky explicitně určit, zda bude tuto kopii pozadí doopravdy potřebovat. SVG prohlížeč může tedy alokovat paměť, jen pokud je to skutečně nutné.

Alokaci paměti zařídí atribut, který se přidá k nadřazenému kontejnerovému prvku (například svg, g, symbol):

- enable-background - možné hodnoty: accumulate (výchozí hodnota), new [x y width height] (rozměry jsou nepovinné a slouží ke zmenšení kopírované a ukládané oblasti pozadí), inherit (zděděná hodnota).

7.7 Parametry osvětlení

SVG filtry umožňují velice flexibilní práci s osvětlením objektů. Základem jsou filtry vkládající do "scény" zdroj světla a popisující jeho specifické vlastnosti, které se následně promítnou do zobrazení objektů.

Vzdálený zdroj světla: "feDistantLight"

Zdánlivě "velmi vzdálený" zdroj světla, jehož paprsky proto mají s dostatečnou přesností v každém bodu grafiky stejný směr.

Atributy:

- ***azimuth*** - ve stupních zapsaný směrový úhel osvětlení v rovině XY.
- ***elevation*** - ve stupních zapsaný směrový úhel osvětlení v rovině YZ.

Bodový zdroj světla: "fePointLight"

Zdánlivě "blízký" zdroj světla, jehož paprsky proto mají v každém bodu grafiky jiný směr.

Atributy:

- ***x, y, z*** - umístění zdroje v 3D prostoru (podle konvence Z souřadnice rostou ve směru pohledu pozorovatele hledícího na grafický obsah).

Zdroj světla typu "spot" (reflektor): "feSpotLight"

U tohoto světla klesá intenzita osvětlení exponenciálně směrem od maximálně osvětleného středu až po vnější omezující úhel.

Atributy:

- ***x, y, z*** - umístění zdroje v 3D prostoru (podle konvence Z souřadnice rostou ve směru pohledu pozorovatele hledícího na grafický obsah).
- ***pointsAtX, pointsAtY, pointsAtZ*** - poloha cíle (bodu), na který je reflektor zaměřen.
- ***specularExponent*** - hodnota exponentu, určujícího zaostření reflektoru; výchozí hodnota se rovná 1.
- ***limitingConeAngle*** - úhel omezující osvětlenou oblast je měřen od osy světelného kužele; pokud tuto hodnotu nedefinujeme, nebude nasvícená oblast nijak omezena. (18)

8. Bitmapové efekty (elementární grafické filtry)

8.1 Přehled elementárních grafických filtrů

Dále uvádím seznam základních filtrů, jejichž kombinací lze sestavit libovolné grafické efekty. SVG norma tyto elementární stavební kameny nazývá **filter primitives**.

Elementární bitmapové filtry

- Filter primitive "**feBlend**" - prolínání dvou bitmap
- Filter primitive "**feComposite**" - slučování dvou bitmap
- Filter primitive "**feMerge**" - slučování dvou bitmap
- Filter primitive "**feComponentTransfer**" - úpravy jasu
- Filter primitive "**feColorMatrix**" - úpravy barevnosti
- Filter primitive "**feConvolveMatrix**" - kombinování sousedních bodů
- Filter primitive "**feFlood**" - vyplnění plochy
- Filter primitive "**feImage**" - vložení další grafiky
- Filter primitive "**feOffset**" - posuv
- Filter primitive "**feDisplacementMap**" - deformace
- Filter primitive "**feTile**" - vyplnění vzorkem
- Filter primitive "**feGaussianBlur**" - rozostření
- Filter primitive "**feMorphology**" - nasílení / zúžení
- Filter primitive "**feTurbulence**" - generování šumu
- Filter primitive "**feSpecularLighting**" - osvětlovací efekt
- Filter primitive "**feDiffuseLighting**" - osvětlovací efekt

- **Definice světelných zdrojů**
  - Light source "**feDistantLight**" - vzdálený zdroj světla
  - Light source "**fePointLight**" - bodové světlo
  - Light source "**feSpotLight**" - reflektor

8.2 Společné atributy

- ***x, y, width, height*** - pomocí těchto nepovinných parametrů lze zúžit oblast pro výpočet konkrétního elementárního filtru
- ***in*** - určuje vstup filtru; možné hodnoty SourceGraphic, SourceAlpha, BackgroundImage, BackgroundAlpha, FillPaint, StrokePaint nebo název obrazové proměnné definované pomocí atributu result v nějakém předcházejícím filtru; pokud tento nepovinný atribut není uveden, použije se výstup bezprostředně předcházející operace (v případě první operace v bloku filter se logicky použije původní grafika = SourceGraphic)
- ***result*** - výsledek operace může být na tomto místě pojmenován, pokud tento atribut není v SVG kódu uveden, je výsledek dané operace ihned předán jako implicitní vstup dalšímu zapsanému filtru

8.3 Definice klíčových slov

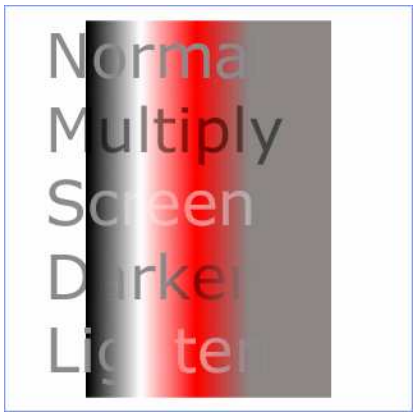
- ***SourceGraphic*** - udává, že vstupem filtrovací operace je původní RGBA grafika (tedy ta skupina, na kterou aplikujeme filtrování) po vykreslení vektorů a případných rastrových obrázků (ale před jakoukoli operací filtrování)
- ***SourceAlpha*** - pouze alfa kanál (tedy průhlednost) původní zdrojové grafiky - viz předchozí bod
- ***BackgroundImage*** - obraz pozadí, viz "Přístup k obrazu pozadí" v článku Kurz SVG - bitmapové efekty (definice W3C, parametry osvětlení)
- ***BackgroundAlpha*** - průhlednost pozadí, viz "Přístup k obrazu pozadí" v článku Kurz SVG - bitmapové efekty (definice W3C, parametry osvětlení)
- ***FillPaint*** - výplň zdrojové grafiky; plocha FillPaint je kompletně neprůhledná (pokud výplň samotná nemá nastavenu průhlednost pomocí atributu fill-opacity)
- ***StrokePaint*** - výplň tahů (obrysů) zdrojové grafiky

8.4 Podrobný popis filtrovacích elementů

Následující část obsahuje **podrobný popis filtrovacích elementů** z normy SVG. Snažím se dále čtenářům předložit u každého jednotlivého grafického filtru názorný příklad, který by co nejlépe osvětlil funkčnost efektu a význam jednotlivých parametrů. Na rozdíl od originální specifikace také nejsou jednotlivá primitiva řazena abecedně, nýbrž jsou seskupena podle podobné funkčnosti.

8.5 Filter primitive "feBlend"

Filtr slučuje dvě různé bitmapy do jedné pomocí prolínacích režimů dobře známých např. z Photoshopu.



Atributy:

- **mode** - režim prolínání, možné hodnoty: normal, multiply, screen, darken, lighten
- **in2** - druhá vstupní bitmapa, význam je obdobou výše zmiňovaného atributu in

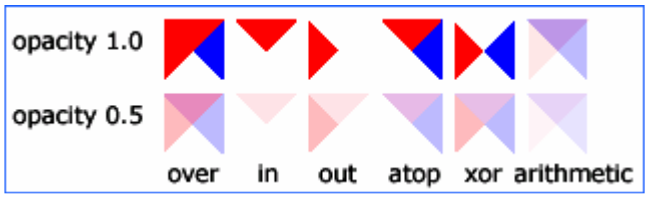
Matematické vyjádření slučování obrazů:

- **význam proměnných**:
  - **qr** - průhlednost výsledku
  - **cr** - výsledná barva (RGB) - přednásobená
  - **qa** - průhlednost obr. A
  - **qb** - průhlednost obr. B
  - **ca** - barva obr. a - přednásobená
  - **cb** - barva obr. B - přednásobená
- **průhlednost pixelů ve výsledku**:
  - **qr** = 1 - (1-qa)*(1-qb)
- **vzorce pro jednotlivé režimy**:
  - ***normal*: cr** = (1-qa)*cb + ca
  - ***multiply*: cr** = (1-qa)*cb + (1-qb)*ca + ca*cb
  - ***screen*: cr** = cb + ca - ca * cb
  - ***darken*: cr** = Min ((1-qa)*cb + ca, (1-qb)*ca + cb)
  - ***lighten*: cr** = Max ((1-qa)*cb + ca, (1-qb)*ca + cb)

Režim *normal* odpovídá standardní metodě slučování v SVG .

8.6 Filter primitive "feComposite"

Filtr ve výsledku kombinuje dvě vstupní bitmapy podle algoritmu Porter-Duff. Možné operace přitom jsou: over, in, atop, out, xor. Jejich význam nejlépe demonstruje vizuální příklad:



V této ukázce se postupně v různých režimech přes modrý trojúhelník kreslí zrcadlově převrácený červený trojúhelník. Upozornit mohu mimo jiné na fakt, že operace "in" je v podstatě maskování (maskou je přitom zmíněný modrý trojúhelník), "out" je rovněž maskovací operace, ale s inverzí masky.

A jak to vypadá v XML?

```
<filter id="in" filterUnits="objectBoundingBox"
  x="-5%" y="-5%" width="110%" height="110%">
  <feFlood flood-color="#ffffff" flood-opacity="1"
result="flood"/>
  <feComposite in="SourceGraphic" in2="BackgroundImage"
    operator="in" result="comp"/>
  <feMerge> <feMergeNode in="flood"/> <feMergeNode in="comp"/>
</feMerge>
</filter>
```

Dále je možná takzvaná aritmetická operace, užitečná například při kombinování výsledků feDiffuseLighting a feSpecularLighting s nějakou texturou.

Matematický zápis aritmetické operace:

- výsledek = k1*i1*i2 + k2*i1 + k3*i2 + k4

Atributy:

- ***operator*** - možné hodnoty: over, in, out, atop, xor, arithmetic
- ***k1, k2, k3, k4*** - konstanty pro výpočet, pokud je operator="arithmetic", jejich výchozí hodnota je vždy 0
- ***in2*** - definuje druhou vstupní bitmapu

8.7 Filter primitive "feMerge"

Jednoduše řečeno provádí tento filtr totéž co **feComposite** v režimu "**over**", jenomže pro libovolný počet bitmapových vrstev. Obecně to bude vypadat takto:

```
<feMerge>
  <feMergeNode in="bitmapa1"/>
  <feMergeNode in="bitmapa2"/>
  <feMergeNode in="..."/>
...
</feMerge>
```

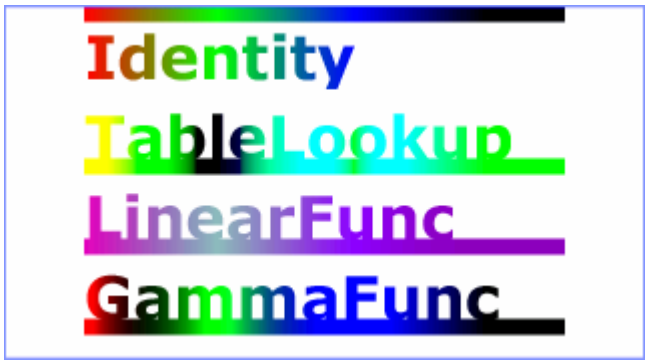
Na uvedeném výpisu názorně vidíte, jak zapsat vícero vnořených prvků **feMergeNode** jejichž atributy in definují jednotlivé slučované vrstvy.

Filter primitive "feComponentTransfer"

Tento filtr přepočítává RGBA data podle následujícího schématu:

- R' = feFuncR(R)
- G' = feFuncG(G)
- B' = feFuncB(B)
- A' = feFuncA(a)

Nabízí se k použití pro dobře známé operace typu úprava jasu, kontrastu a podobně. Na rozdíl od většiny *pracuje s nepřednásobenými hodnotami RGB.*



Podle typu zvolené operace v atributu type bude přirozeně nutné přidat další atributy s parametry:

1. Pro funkci **identity**:
   C' = C (C zde samozřejmě zastupuje kompozitní barvu)
2. Pro funkci **table**: Lineární interpolace podle tabulkových hodnot uvedených v atributu tableValues.
   Interpolační vzorec je definován následujícím způsobem. Tabulka obsahuje (N+1) hodnot. Pro hodnotu **C** z **k**-tého intervalu tabulky vypočteme výslednou barvu **C'** takto:
   $C' = table_k + (C-k/N)*(table_{k+1}-table_k)$
   (vzorec předpokládá hodnoty barev **C** v rozsahu 0..1 a indexu k v rozsahu 0..N)
3. Pro funkci **discrete**: Vše je v podstatě shodné s operací **table**, pouze nedojde k interpolaci:
   $C' = table_k$
4. Pro funkci **linear** je výpočet následující:
   C' = slope * C + intercept
5. Pro funkci **gamma** je to exponenciála:
   C' = amplitude * pow(C, exponent) + offset
   (pow = mocnina)

Atributy:

- ***type*** - typ funkce, možné hodnoty: identity, table, discrete, linear, gamma; podle typu vybrané funkce budou potřeba některé z následujících atributů
- ***tableValues*** - seznam hodnot oddělených mezerami nebo čárkami
- ***slope*** - strmost přímky, výchozí hodnota se rovná 1
- ***intercept*** - posun přímky na svislé ose funkce
- ***amplitude*** - amplituda gama funkce, výchozí hodnota 1
- ***exponent*** - exponent gama funkce, výchozí hodnota 1
- ***offset*** - "svislý" posun gama funkce

Nyní už vám pochopení ukázkového SVG kódu jistě nebude činit problémy:

```
<?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 width="320px" viewBox="0 0 800 440"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <title>
  feComponentTransfer - operace s& tabulkou barev - www.interval.cz
  </title>
  <desc>
  Pomoci tohoto filtru muzete realizovat operace ktere jsou
  ve Photoshopu k nalezeni pod nabidkou "Image/Adjust/Levels"
  resp. "Curves".
  </desc>
  <defs>
    <linearGradient id="MyGradient" gradientUnits="userSpaceOnUse"
      x1="100" y1="0" x2="600" y2="0">
      <stop offset="0" stop-color="#ff0000" />
      <stop offset=".33" stop-color="#00ff00" />
      <stop offset=".67" stop-color="#0000ff" />
      <stop offset="1" stop-color="#000000" />
    </linearGradient>
    <filter id="Identity" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
      <feComponentTransfer>
        <feFuncR type="identity"/>
        <feFuncG type="identity"/>
        <feFuncB type="identity"/>
        <feFuncA type="identity"/>
      </feComponentTransfer>
    </filter>
    <filter id="Table" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
      <feComponentTransfer>
        <feFuncR type="table" tableValues="0 0 1 1"/>
        <feFuncG type="table" tableValues="1 1 0 0"/>
        <feFuncB type="table" tableValues="0 1 1 0"/>
      </feComponentTransfer>
   </filter>
   <filter id="Linear" filterUnits="objectBoundingBox"
           x="0%" y="0%" width="100%" height="100%">
     <feComponentTransfer>
       <feFuncR type="linear" slope=".5" intercept=".25"/>
       <feFuncG type="linear" slope=".5" intercept="0"/>
       <feFuncB type="linear" slope=".5" intercept=".5"/>
     </feComponentTransfer>
   </filter>
   <filter id="Gamma" filterUnits="objectBoundingBox"
           x="0%" y="0%" width="100%" height="100%">
     <feComponentTransfer>
       <feFuncR type="gamma" amplitude="2" exponent="5" offset="0"/>
       <feFuncG type="gamma" amplitude="2" exponent="3" offset="0"/>
       <feFuncB type="gamma" amplitude="2" exponent="1" offset="0"/>
     </feComponentTransfer>
    </filter>
  </defs>
 <g font-family="Verdana" font-size="75"
   font-weight="bold" fill="url(#MyGradient)" >
   <rect x="100" y="0" width="600" height="20" />
   <text x="100" y="90" filter="url(#Identity)">Identity</text>
   <text x="100" y="190" filter="url(#Table)" >TableLookup</text>
   <rect x="100" y="190" width="600" height="20" filter="url(#Table)" />
   <text x="100" y="290" filter="url(#Linear)" >LinearFunc</text>
   <rect x="100" y="290" width="600" height="20" filter="url(#Linear)" />
    <text x="100" y="390" filter="url(#Gamma)" >GammaFunc</text>
    <rect x="100" y="390" width="600" height="20" filter="url(#Gamma)" />
  </g>
  <rect fill="none" stroke="blue" x="1" y="1" width="798" height="438"/>
</svg>
```

8.8 Filter primitive "feColorMatrix"

Pomocí níže uvedené matematické maticové operace se realizují všem grafikům jistě důvěrně známé operace typu změna saturace barev nebo třeba posuv barevného spektra...

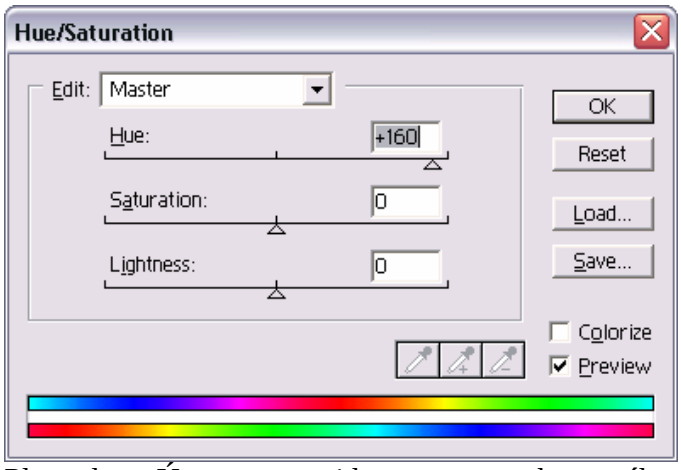
$$\begin{pmatrix} R' \\ G' \\ B' \\ A' \\ 1 \end{pmatrix} = \begin{pmatrix} a00 & a01 & a02 & a03 & a04 \\ a10 & a11 & a12 & a13 & a14 \\ a20 & a21 & a22 & a23 & a24 \\ a30 & a31 & a32 & a33 & a34 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \\ A \\ 1 \end{pmatrix}$$

Matice pro výpočet zobrazení filtru **feComponentTransfer**

Při výpočtu se tentokráte pracuje s nepřednásobenými hodnotami barev.

Možná je i extrakce jednotlivých barvových kanálů

A tohle jistě znáte z praxe...



Photoshop: Úpravy sytosti barev a posuv barevného spektra

Atributy:

- **type** - uvádí typ operace: matrix (zcela obecný výpočet, kdy je nutno udat všech 5x4 hodnot výpočetní matice), saturate (změna saturace neboli sytosti barev), hueRotate (posuv barevného spektra), luminanceToAlpha (hodnota průhlednosti je odvozena z jasu příslušného bodu)
- **values** - atribut obsahuje seznam hodnot nutných pro příslušný typ výpočtu:
    - pro "***matrix***" - všech 5x4 hodnot matice (a00 a01 a02 a03 a04 a10 a11 ... a34)
    - pro "***saturate***" - jedno reálné číslo v rozsahu 0...1
    - pro "***hueRotate***" - jedno reálné číslo (úhlové stupně)
    - pro "***luminanceToAlpha***" - není potřeba žádná hodnota

XML kód ukázky:

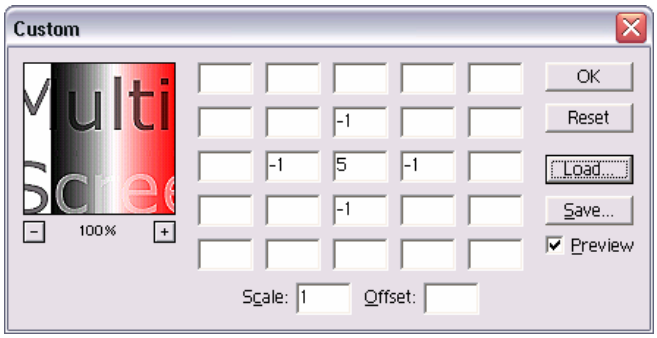
```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
          "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="320px" viewBox="0 0 800 640"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <title>
  feColorMatrix - priklady barvove korekce - www.interval.cz
  </title>
  <desc>
  Pomoci tohoto filtru muzete realizovat operace ktere jsou
  ve Photoshopu k nalezeni pod nabidkou "Image/Adjust/Hue/Saturation".
  Posledni 2 priklady ukazuji dalsi mozne vyuziti obecne matice -
  extrakce jednotlivych barvovych kanalu.
  </desc>
  <defs>
    <linearGradient id="MyGradient" gradientUnits="userSpaceOnUse"
      x1="100" y1="0" x2="500" y2="0">
      <stop offset="0" stop-color="#f00" />
      <stop offset=".33" stop-color="#0f0" />
      <stop offset=".67" stop-color="#00f" />
      <stop offset="1" stop-color="#000" />
    </linearGradient>
    <filter id="MatrixRed" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
      <feColorMatrix type="matrix" in="SourceGraphic"
           values=".99 .00 .00 0 0
                   .00 .00 .00 0 0
                   .00 .00 .00 0 0
.00 .00 .00 1 0"/>
    </filter>
    <filter id="MatrixGreen" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
      <feColorMatrix type="matrix" in="SourceGraphic"
           values=".00 .00 .00 0 0
                   .00 .99 .00 0 0
                   .00 .00 .00 0 0
.00 .00 .00 1 0"/>
    </filter>
```

```
    <filter id="Saturate" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
      <feColorMatrix type="saturate" in="SourceGraphic" values="30%"/>
    </filter>
    <filter id="HueRotate90" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">

 <feColorMatrix type="hueRotate" in="SourceGraphic" values="90"/>
    </filter>
    <filter id="LuminanceToAlpha" filterUnits="objectBoundingBox"
            x="0%" y="0%" width="100%" height="100%">
<feColorMatrix type="luminanceToAlpha" in="SourceGraphic"/>
     </filter>
  </defs>
  <rect fill="none" stroke="blue"
        x="1" y="1" width="798" height="638"/>
  <g font-family="Verdana" font-size="75"
     font-weight="bold" fill="url(#MyGradient)" >
    <rect x="100" y="0" width="500" height="20" />
    <text x="100" y="90">Unfiltered</text>
    <text x="100" y="190" filter="url(#Saturate)" >Saturate</text>
    <rect x="100" y="190" width="500" height="20" filter="url(#Saturate)" />
    <text x="100" y="290" filter="url(#HueRotate90)" >HueRotate</text>
    <rect x="100" y="290" width="500" height="20" filter="url(#HueRotate90)"
/>
    <text x="100" y="390" filter="url(#LuminanceToAlpha)" >Luminance</text>
    <rect x="100" y="390" width="500" height="20"
filter="url(#LuminanceToAlpha)" />
    <text x="100" y="490" filter="url(#MatrixRed)" >R-channel</text>
    <rect x="100" y="490" width="500" height="20" filter="url(#MatrixRed)"
/>
    <text x="100" y="590" filter="url(#MatrixGreen)" >G-channel</text>
    <rect x="100" y="590" width="500" height="20" filter="url(#MatrixGreen)"
/>
  </g>
</svg>
```

8.9 Filter primitive "feConvolveMatrix"

Pro dosažení výsledku tento filtr navzájem kombinuje sousedící obrazové body. Mezi efekty, kterých lze dosáhnout tímto postupem, patří například rozostření a zostření, detekce hran (blurring či sharpening, edge detection, embossing, beveling) a další.



Ekvivalent SVG filtru **feConvolveMatrix** ve Photoshopu. Jedná se o jednu ze základních operací bitmapové grafiky na níž je postavena spousta dalších komplexnějších efektů.

Filtr lze matematicky popsat jako aplikaci konvoluční matice na zdrojovou bitmapu. Toto je schematický zápis příslušného algoritmu:

$$
\begin{aligned}
&\text{VÝSTUP}_{X,Y} = (\\
&\quad \text{SUMA(I=0 to [orderY-1]) } \{\\
&\qquad \text{SUMA(J=0 to [orderX-1]) } \{\\
&\qquad\quad \text{in}_{\text{X-targetX+J, Y-targetY+I}}\\
&\qquad\quad * \text{kernelMatrix}_{\text{orderX-J-1, orderY-I-1}}\\
&\qquad \}\\
&\quad \}\\
&) \,/\, \text{divisor} + \text{bias}
\end{aligned}
$$

Atributy:

- ***order*** - rozměr konvoluční matice, volitelně lze uvést i druhou hodnotu - pokud matice není čtvercová
- ***kernelMatrix*** - vlastní obsah konvoluční matice (kernelu) zapsán po řádcích
- divisor - dělitel je součtem všech členů matice a je nepovinný (součet bude prohlížečem SVG grafiky doplněn automaticky; pro případ, kdy by byl roven 0, bude změněn na hodnotu 1)
- ***bias*** - konstanta, která se přičte k výsledné hodnotě (výhodná může být např. v když je třeba aby prázdný (nulový) výstup byl 50% šedá barva)
- ***targetX*** - je (nepovinná) relativní X poloha konvoluční matice vůči pixelu, pro který zrovna probíhá výpočet; sloupec matice umístěný co nejvíce vlevo má číslo 0; výchozí hodnotou atributu je umístění matice na střed
- ***targetY*** - je (nepovinná) relativní Y poloha konvoluční matice vůči pixelu, pro který zrovna probíhá výpočet; nejhořejší sloupec matice má číslo 0; výchozí hodnotou atributu je umístění matice na střed

- ***edgeMode*** - určuje způsob výpočtu na okrajích obr. rastru, kde samozřejmě některé obrazové body nutné pro výpočet neexistují; možné hodnoty s významem uvedeným v závorkách jsou: duplicate (zopakují se okrajové body), wrap (vezmou se hodnoty z opačného okraje bitmapy - možná ještě lepší je představit si, jako by se k hraně, kde nám chybí pixely přidala ještě jedna kopie obrazového rastru - podobně jako se skládají vedle sebe dlaždice) a none (použijí se nulové - černé hodnoty)
- ***kernelUnitLength*** - aby byla zajištěna škálovatelnost a nezávislost filtru na rozlišení, je dobré zde nastavit, jak velká je vlastně jedna buňka matice; lze uvést jednu nebo dvě hodnoty (pokud se měřítko ve směru Y liší); jestliže hodnotu nezapíšete, SVG prohlížeč předpokládá, že jedna buňka konvoluční matice odpovídá jednomu pixelu obrázku, což je možná rychlejší na výpočet, ale výsledek se bude lišit podle rozlišení výstupního zařízení
- ***preserveAlpha*** - pokud se nastaví na hodnotu "true" bude zachován kanál průhlednosti, výchozí hodnota "false" způsobí aplikaci filtru na všechny čtyři obrazové kanály (24)

SVG kód příkladu následuje:

```
<?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 width="320px" viewBox="0 0 320 120"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <title>feConvolveMatrix - www.interval.cz</title>
<desc>
Mimo to, ze se zde demonstruje funkcnost vyse uvedeneho filtru,
upozornuji na CHYBEJICI atribut "kernelUnitLength". Pokud jej
neuvedete, bude vypocet fitru rychlejsi, ale relativni velikost
efektu se bude menit v zavislosti na meritku zobrazeni.
(Vyzkousejte "ZOOM" v SVG prohlizeci!)
  </desc>
  <defs>
    <filter id="motionBlur" filterUnits="objectBoundingBox"
      x="0%" y="-3%" width="120%" height="106%">
<feConvolveMatrix in="SourceGraphic" edgeMode="wrap" order="11,1"
kernelMatrix="0.2 0.5 1 2 3 4 5 6 7 8 22"/>
    </filter>
  </defs>
<rect fill="white" stroke="blue" x="1" y="1" width="318"
height="118"/>
<text x="20" y="32" font-size="36px" filter="url(#motionBlur)">
    feConvolveMatrix
  </text>
<image xlink:href="svg.gif" x="20" y="40" width="80" height="31"/>
<image xlink:href="svg.gif" x="20" y="80" width="80" height="31"
filter="url(#motionBlur)"/>
</svg>
```

8.10 Filter primitive "feFlood"

Vyplní oblast filtru konstantní barvou a průhledností.

Atributy:

- **flood-color** - barva výplně, krom uvedení barvy lze použít klíčové slovo "currentColor" (umožňuje použití barvy nastavené v atributu "color", což je dobré například pro převzetí barvy nastylované v nadřazeném XHTML kódu); výchozí hodnota: black
- **flood-opacity** - průhlednost výplně; výchozí hodnota: 1

```
<feFlood flood-color="white" flood-opacity="1" result="flood"/>
```

8.11 Filter primitive "feImage"

Vložení obrazu do pracovní oblasti filtru. Vkládaným obrazem může být libovolný vektorový objekt v rámci SVG souboru, ale podporováno je i externí vkládání formátů JPEG, PNG a dokonce i SVG samotného. Posledně jmenovanou variantu v současnosti zatím příliš nedoporučuji, jelikož s ní mnoho současných prohlížečů - díky obtížnosti její implementace - bude mít problémy.

Chování filtru a jeho implementace by měla být shodná s prvkem "image" pro případ externího odkazu. Pro případ interního odkazu bude chování shodné s elementem "use".

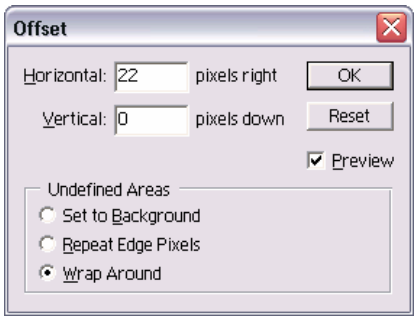
Atributy:

- **xlink:href** - odkaz na externí obraz nebo též na pojmenovaný objekt uvnitř SVG souboru

```
<feImage xlink:href="#vectorObject" result="bitmap"/>
```

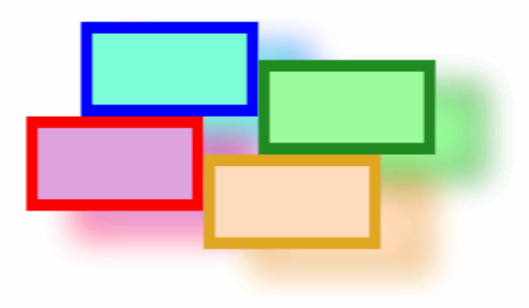
8.12 Filter primitive "feOffset"

Filtr posouvá bitmapový rastr ve směru X nebo Y. Tato operace se očividně nabízí například k výrobě oblíbených stínů...



Photoshop: Tuto funkci najdete v menu pod "Filter/Other/Offset"

Pokud souřadnicový systém, ve kterém filtr pracuje, neodpovídá pixelům výstupního zařízení, je dobré si uvědomit, že může dojít k přepočtům (interpolaci) obrazu. Při takové interpolaci se (většinou zbytečně) zvyšuje časová a výkonová spotřeba filtru. Takže doporučuji používat atribut image-rendering ovlivňující interpolaci bitmapových obrazů a filtrů - a nastavit jej na maximální rychlost: image-rendering="optimizeSpeed".



Atributy:

- ***dx*** - posun ve směru osy X (připomínám, že se opět jedná o aktuálně nastavený souřadnicový systém, jak je v SVG dobrým zvykem)
- ***dy*** - posun ve směru osy Y

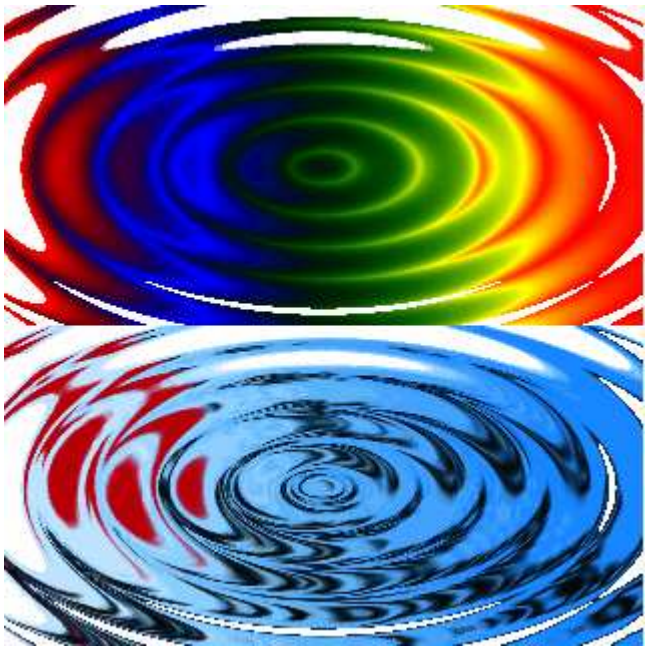
Výpis XML kódu:

```
<?xml version="1.0"?>
<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">
  <title>feOffset - priklad pouziti k vyrobe stinu -
www.interval.cz</title>
  <desc>
    Povsimnete si prosim nastaveni rozmeru u prvku
    "filter" - vzdy se snazime o jejich minimalizaci!
  </desc>
  <filter id="shadow30_15"
      x="-2%" y="-4%" width="128%" height="128%">
    <feOffset in="SourceGraphic" dx="30" dy="15" />
    <feGaussianBlur result="blur" stdDeviation="10" />
    <feBlend in="SourceGraphic" in2="blur" mode="normal" />
  </filter>
  <g stroke-width="7" filter="url(#shadow30_15)" image-
rendering="optimizeSpeed">
    <rect x="53" y="10" width="100" height="50"
      stroke="blue" fill="aquamarine" />
    <rect x="160" y="33" width="100" height="50"
      stroke="forestgreen" fill="palegreen" />
    <rect x="20" y="67" width="100" height="50"
      stroke="red" fill="plum" />
    <rect x="127" y="90" width="100" height="50"
      stroke="goldenrod" fill="peachpuff" />
  </g>
</svg>
```

8.13 Filter primitive "feDisplacementMap"

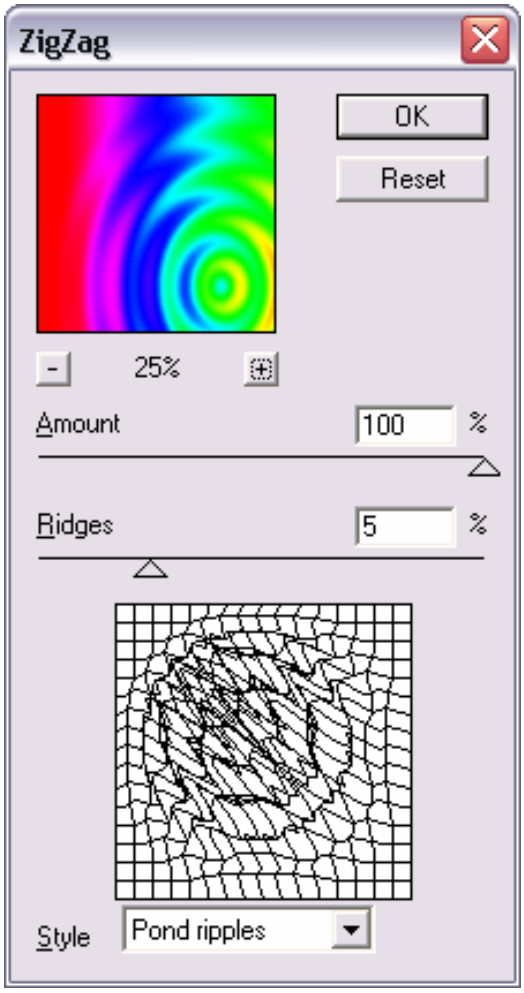
Filtr získá výsledek tak, že používá hodnoty pixelů ze zdroje in2 k deformaci (posouvání) jednotlivých obrazových bodů na vstupu in. Mezi praktická využití je možné zařadit kupříkladu simulaci lomu světla na trojrozměrném objektu (zdeformování odrazu objektu na zvlněné vodní hladině nebo obraz viděný přes tvarované sklo).

Ačkoli to tak nemusí na první pohled vypadat, patří tento filtr mezi nejzajímavější a nejvíce "kreativní" grafické funkce. Pomocí něho lze realizovat v podstatě všechny filtry Photoshopu ze skupiny "Distort" (Deformace).



Kouzlo vězí v použití šikovné deformační mapy (displace map). Spoustu takových užitečných vzorků můžete najít ve všech verzích Photoshopu v následujících složkách (podadresářích): \Photoshop\Plug-ins\Displacement maps\, \Photoshop\Presets\Textures\, \Photoshop\Presets\Patterns\.

```
<?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 width="320px" viewBox="0 0 320 320"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <title>
  feDisplacementMap - efekt "ZigZag" - www.interval.cz
  </title>
  <desc>
  Bitmapu definujici "rozvlneni" si tomto pripade
  vytvorime umele z kruhoveho prechodu barev.
  Vysledkem je efekt znamy z Photoshopu jako "ZigZag".
  </desc>
<defs>
  <!-- MH: tento kruhovy prechod vytvori "displace" mapu pro
deformacni efekt -->
  <radialGradient id='gradient' spreadMethod='reflect'
    gradientUnits='objectBoundingBox'
    cx='50%' cy='50%' r='5%' fx='50%' fy='50%'>
    <stop id='c1' offset='5%' stop-color='red' stop-opacity='1'/>
    <stop id='c2' offset='95%' stop-color='yellow' stop-
opacity='1'/>
  </radialGradient>
  <!-- MH: na tomto prechodu si ukazeme jak muze byt vysledek
pusobivy -->
  <linearGradient id='gr' gradientUnits='objectBoundingBox'
    x1='0%' y1='0%' x2='100%' y2='0%'>
    <stop offset='00%' stop-color='red' stop-opacity='1'/>
    <stop offset='25%' stop-color='blue' stop-opacity='1'/>
    <stop offset='50%' stop-color='green' stop-opacity='1'/>
    <stop offset='75%' stop-color='yellow' stop-opacity='1'/>
    <stop offset='99%' stop-color='red' stop-opacity='1'/>
  </linearGradient>
  <g id='displaceMap'>
    <rect x='0' y='000' width='320' height='160'
fill='url(#gradient)'/>
    <rect x='0' y='160' width='320' height='160'
fill='url(#gradient)'/>
  </g>
  <filter id='ZigZag44' filterUnits='objectBoundingBox'
    x='0%' y='0%' width='100%' height='100%'>
    <!-- MH: zde si vytvorime "displace" mapu z vektoroveho
objektu -->
    <!-- MH: (viz id="displaceMap") vyplneneho kruhovym prechodem
-->
    <feImage xlink:href='#displaceMap' result='map'/>
    <!-- MH: "scale" urcuje velikost efektu -->
    <feDisplacementMap scale='44'
      xChannelSelector='G' yChannelSelector='G'
      in2='map' in='SourceGraphic'/>
  </filter>
</defs>
<!-- MH: Tady jsou dve ukazky. Velmi efektni efekt. -->
<rect x='0' y='0' width='320' height='160'
  fill='url(#gr)' filter='url(#ZigZag44)'/>
<image
x='0' y='160' width='320' height='160'
  xlink:href='svg.gif' filter='url(#ZigZag44)'/>
</svg>
```



Photoshop: Jeden z mnoha efektů dosažitelných pomocí "displace map" ..

I tento filtr je ovlivněn nastavením atributu image-rendering.

Atributy:

- ***scale*** - měřítko deformace (výchozí hodnota je 0)
- ***in2*** - druhá vstupní bitmapa, která určuje deformace obrazu v in
- ***xChannelSelector*** - určuje, který barvový kanál z in2 bude určovat deformace ve směru X; možné hodnoty: R, G, B, A
- ***yChannelSelector*** - určuje, který barvový kanál z in2 bude určovat deformace ve směru Y; možné hodnoty: R, G, B, A

8.14 Filter primitive "feTile"

Vyplní cílovou plochu opakujícími se kopiemi zdrojové bitmapy. Jinými slovy je to obdoba výplně pomocí vzorku či textury.

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

Je logické, že tato operace má význam pro případ, kdy je vstupní obraz menší než výstupní.

```
<?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 width="320px" viewBox="0 0 320 240"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <title>feTile - vykresleni textury - www.interval.cz</title>
  <defs>
     <filter id="tile" filterUnits="userSpaceOnUse">
       <feImage xlink:href="svg.gif"
         x="8" y="0" width="88" height="31" result="img"/>
       <feTile x="0" y="0" width="320" height="210" in="img"/>
     </filter>
  </defs>
  <rect x="0" y="0" width="320" height="240" filter="url(#tile)"/>
</svg>
```

8.15 Filter primitive "feGaussianBlur"

Gausovské rozostření - jeden z naprosto základních a nejpoužívanějších efektů vůbec.

Klasické gausovské rozostření může mít v SVG pro každou osu jinou velikost.

Prakticky používaná výpočetní konvoluční matice je aproximací výrazu:

$H(x) = \exp(-x^2 / (2*s^2)) / \sqrt{2*pi*s^2}$

Zde s je standardní odchylka nastavená v atributu stdDeviation.

Atributy:

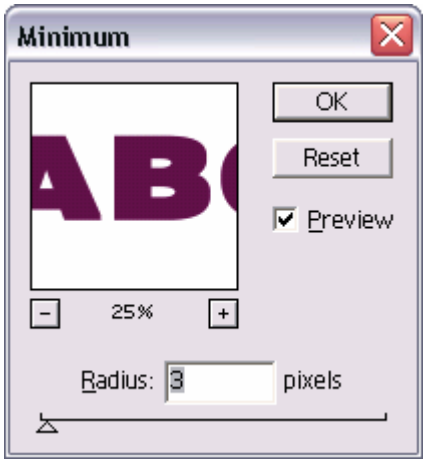
- ***stdDeviation*** - standardní odchylka >= 0 (z praktického hlediska řekněme velikost rozostření), uvedeny mohou být dvě hodnoty, pro každou souřadnicovou osu zvlášť; pozor, hodnota 0 vypne filtr a na výstupu se potom objeví 0, tedy průhledná černá

```
<filter id="blurX" filterUnits="objectBoundingBox"
  x="-25%" y="-3%" width="150%" height="106%">
    <!-- "stdDeviation" controls size of blur effect (X and Y) -->
    <feGaussianBlur stdDeviation="201"/>
</filter>
```

8.16 Filter primitive "feMorphology"

Provádí "ztučnění" nebo naopak "ztenčení" nakresleného obrázku. Nejčastěji jej nejspíše využijete pro úpravy alfa kanálu.

Photoshop: Ekvivalent SVG filtru "**feMorphology**"



Atributy:

- ***operator*** - možné hodnoty: erode (zúžení), dilate (rozšíření)
- ***radius*** - poloměr operace (nebo dva poloměry - pro druhou souřadnicovou osu totiž může být hodnota jiná); výchozí hodnota 0 filtr vypne, tj. výsledkem bude průhledná černá

Zápis vypadá v praxi takto:

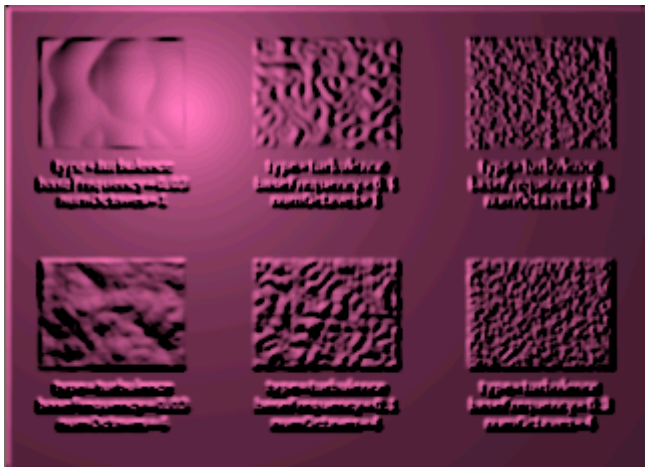
```
<filter id="Dilate3">
  <feMorphology operator="dilate" in="SourceGraphic" radius="4" />
</filter>
```

8.17 Filter primitive "feTurbulence"

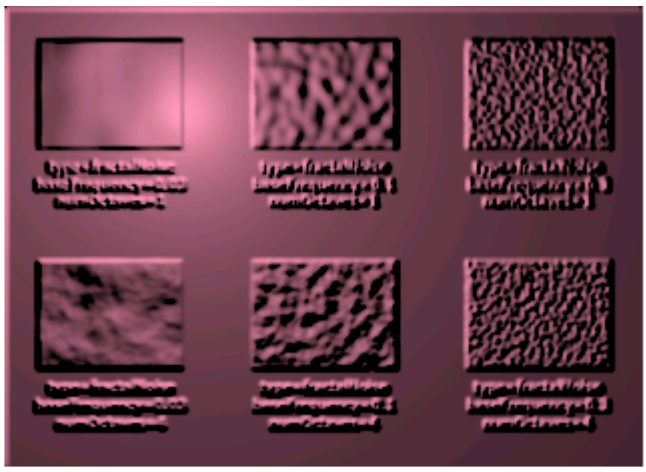
Filtr generuje obraz pomocí takzvané **Perlin turbulence** funkce. Tu lze použít k syntéze umělých textur napodobujících přírodu, například šumu, oblak nebo mramoru.

Příklad zápisu:

Velmi zajímavé výstupy například dosáhnete, použijete-li feTurbulence k vygenerování **bumpmapy** pro filtr feDiffuseLighting. Tuto možnost demonstrují následující příklady.



Filtr v režimu "**turbulence**" v ukázce funguje jako "**bumpmap**" pro osvětlovací filtr.



Filtr v režimu "**fractalNoise**" zde funguje jako "**bumpmap**" pro osvětlovací filtr

Atributy:

- ***type*** - typ funkce určuje zda se bude generovat šum (fractalNoise) nebo víření (turbulence)
- ***baseFrequency*** - základní kmitočet šumové funkce, pokud jsou zapsány dvě hodnoty, tak se druhá uplatní pro směr osy Y, výchozí hodnota je 0
- ***numOctaves*** - druhý parametr šumové funkce, výchozí hodnota 1
- seed - počáteční startovací číslo pro generátor pseudonáhodných čísel, výchozí hodnota je 0
- ***stitchTiles*** - tento parametr přijde ke slovu pokud bychom chtěli bitmapu vygenerovanou touto funkcí využít ke skládání větší textury (pomocí filtru feTile); možné hodnoty atributu: noStitch (výchozí), stitch (pokud je nastavena tato hodnota, tak prohlížeč automaticky přizpůsobí některé parametry - jako třeba kmitočet - aby se barvy na protilehlých okrajích shodovaly a dosáhlo se hladkého, bezešvého napojení na okrajích jednotlivých dílků)

8.18 Filter primitive "feDiffuseLighting"

Tento a následující filtr feSpecularLighting používá Phongův osvětlovací model k simulaci reálného osvětlení skutečného plastického (3D) povrchu. Teoretické podrobnosti naleznete ve všech učebnicích počítačové grafiky (například "Algoritmy počítačové grafiky"; Sochor J., Žára J., Beneš B.; Vydavatelství ČVUT, 1996; kap. 11.1.1). Detaily praktické realizace v SVG prohlížečích obsahuje specifikace SVG.

Pomocí tohoto efektu lze mimo jiné získat takové ty úžasné plastické objekty s realistickými světelnými odlesky.

Bodový zdroj světla

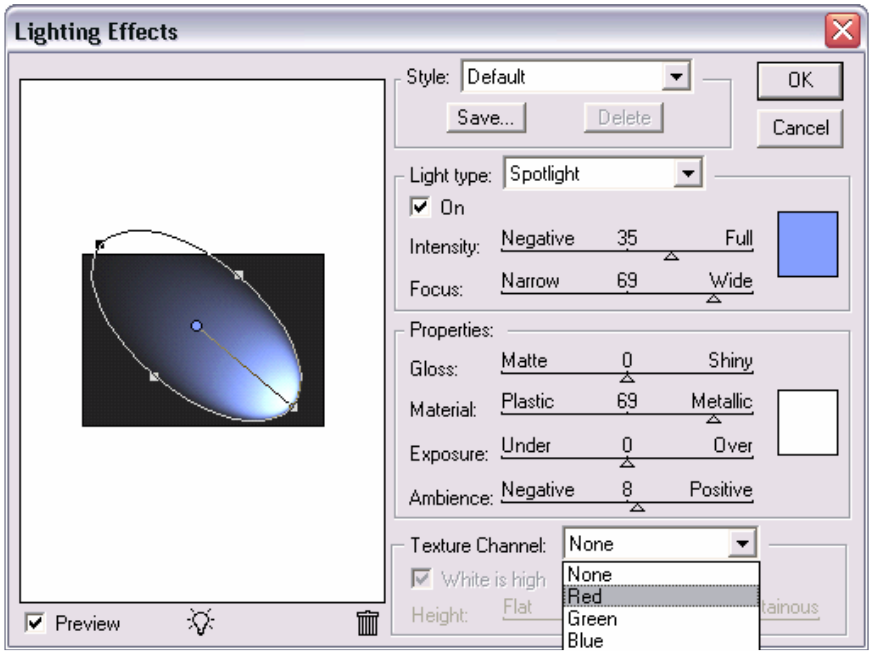
Zápis XML:

```
<filter id="diffPoint" x="0%" y="0%" width="100%" height="100%">
  <feGaussianBlur stdDeviation="1"/>
  <feDiffuseLighting surfaceScale="3" lighting-color="#BFB"
    diffuseConstant="1">
    <fePointLight x="125" y="75" z="25"/>
  </feDiffuseLighting>
</filter>
```

Světelný zdroj typu reflektor.

Zápis XML:

```
<filter id="diffSpot" x="0%" y="0%" width="100%" height="100%">
  <feGaussianBlur stdDeviation="1" />
  <feDiffuseLighting surfaceScale="3" lighting-color="#FFB"
    diffuseConstant="1">
    <feSpotLight x="0" y="0" z="120"
      pointsAtX="125" pointsAtY="150" pointsAtZ="-30"
      specularExponent="44" limitingConeAngle="90"/>
  </feDiffuseLighting>
</filter>
```



Odpovídající filtr ve Photoshopu pod názvem "Lighting Effects" (Světelné efekty)

Atributy:

- ***surfaceScale*** - měřítko určující velikost nerovností osvětlovaného povrchu (bumpmap uložené v alfa kanálu na vstupu in), výchozí hodnota 1
- ***diffuseConstant*** - difusní konstanta (větší než 0) pro odraz světla v Phongově modelu, výchozí hodnota 1
- ***kernelUnitLength*** - aby byla zajištěna škálovatelnost a nezávislost filtru na rozlišení, je dobré zde nastavit, jak velká je vlastně jedna buňka matice; lze uvést jednu nebo dvě hodnoty (pokud se měřítko ve směru Y liší); jestliže hodnotu nezapíšete, SVG prohlížeč předpokládá, že jedna buňka konvoluční matice odpovídá jednomu pixlu obrázku, což je možná rychlejší na výpočet, ale výsledek se bude lišit podle rozlišení výstupního zařízení

8.19 Filter primitive "feSpecularLighting"

Tento a předcházející filtr feDiffuseLighting používá Phongův osvětlovací model k simulaci nasvícení trojrozměrného povrchu. Předcházející filtr feDiffuseLighting počítá difusní složku (stínování) Phongova modelu, zatímco feSpecularLighting počítá takzvanou "specular" složku (to jsou v podstatě odlesky).

Bodový zdroj světla.

Zápis XML:

```
<filter id="specPoint" x="0%" y="0%" width="100%" height="100%">
  <feGaussianBlur stdDeviation="1"/>
  <feSpecularLighting surfaceScale="3" lighting-color="#BFB"
    specularConstant="3" specularExponent="9">
    <fePointLight x="125" y="75" z="25"/>
  </feSpecularLighting>
</filter>
```

Světelný zdroj typu reflektor

Zápis XML:

```
<filter id="specSpot" x="0%" y="0%" width="100%" height="100%">
  <feGaussianBlur stdDeviation="1" />
  <feSpecularLighting surfaceScale="3" lighting-color="#FFB"
    specularConstant="3" specularExponent="9">
    <feSpotLight x="0" y="0" z="120"
      pointsAtX="125" pointsAtY="150" pointsAtZ="-30"
      specularExponent="44" limitingConeAngle="90"/>
  </feSpecularLighting>
</filter>
```

Atributy:

- ***surfaceScale*** - měřítko určující velikost nerovností osvětlovaného povrchu (bumpmap uložené v alfa kanálu na vstupu in), výchozí hodnota 1
- ***specularConstant*** - "specular" konstanta (větší než 0) pro odraz světla v Phongově modelu, výchozí hodnota 1
- ***specularExponent*** - "specular" exponent v Phongově matematickém výrazu, platí: čím vyšší hodnota, tím vyšší lesklost materiálu virtuálního povrchu, povolený rozsah hodnot v SVG: od 1.0 do 128.0, výchozí hodnota 1
- ***lighting-color*** - definuje barvu světla u všech třech typů osvětlení. Nabývá možných hodnot: currentColor a dále všech možných RGB hodnot povolených v SVG specifikaci; výchozí hodnota: white (bílá)
- ***kernelUnitLength*** - aby byla zajištěna škálovatelnost a nezávislost filtru na rozlišení, je dobré zde nastavit, jak velká je vlastně jedna buňka matice; lze uvést jednu nebo dvě hodnoty (pokud se měřítko ve směru Y liší); jestliže hodnotu nezapíšete, SVG prohlížeč předpokládá, že jedna buňka konvoluční matice

odpovídá jednomu pixelu obrázku, což je možná rychlejší na výpočet, ale výsledek se bude lišit podle rozlišení výstupního zařízení

8.20 Phongův vzorec ve 2D

Phongův model počítá osvětlení v trojrozměrném prostoru, ale my se pohybujeme v oblasti 2D grafiky. Nikterak překvapivě je proto osvětlovaný povrch umístěn do roviny X-Y se souřadnicí Z=0. Podmínkou je, že pozorovatel je jakoby "velmi daleko" od zobrazeného povrchu, takže s dostatečnou přesností lze předpokládat, že jednotkový vektor ve směru jeho pohledu je v každém bodu (0,0,1).

Plasticita povrchu je simulována pomocí **bumpmap**, což je šedotónový alfa kanál a hodnota šedé v konkrétním bodu určuje virtuální výšku povrchu. (Ve Photoshopu je **bumpmap** nazývána "Texture Channel".)

8.21 Zdroj světla

Filtr může počítat osvětlení způsobené **jedním** světelným zdrojem, který je definován pomocí vnořeného (synovského) objektu. a to **feDistantLight**, **fePointLight** nebo **feSpotLight**.

8.22 Zpracování výsledku a více světelných zdrojů

Filtr feDiffuseLighting počítá difusní složku (stínování) Phongova modelu, zatímco následující feSpecularLighting počítá takzvanou "specular" složku (to jsou v podstatě odlesky).

Výsledná bitmapa obsahující nasvícení povrchu je po výpočtu vždy kompletně vyplněna. Alfa kanál výsledku přitom obsahuje - řekněme - **váhu** nasvícení pro každý bod bitmapy.

Nejvhodnější metodou, jak tuto světelnou mapu dále zpracovat, je sloučit ji s podkladem (texturou) pomocí filtrů feBlend nebo feComposite v režimu "arithmetic". **Více zdrojů světla** lze velmi snadno přidat vícenásobným výpočtem podle Phongova modelu a sečtením jednotlivých světelných map ještě před sloučením s pozadím. (25)

9 Definice světelných zdrojů

9.1 Vzdálený zdroj světla: "feDistantLight"

Zdánlivě "velmi vzdálený" zdroj světla, jehož paprsky proto mají s dostatečnou přesností v každém bodu grafiky stejný směr.

Atributy:

- ***azimuth*** - ve stupních zapsaný směrový úhel osvětlení v rovině XY.
- ***elevation*** - ve stupních zapsaný směrový úhel osvětlení v rovině YZ.

9.2 Bodový zdroj světla: "fePointLight"

Zdánlivě "blízký" zdroj světla, jehož paprsky proto mají v každém bodu grafiky jiný směr.

Atributy:

- **x, y, z** - umístění zdroje v 3D prostoru (podle konvence z souřadnice rostou ve směru pohledu pozorovatele hledícího na grafický obsah).

9.3 Zdroj světla typu "spot" (reflektor): "feSpotLight"

U tohoto světla klesá intenzita osvětlení exponenciálně směrem od maximálně osvětleného středu až po vnější omezující úhel.

Atributy:

- ***x, y, z*** - umístění zdroje v 3D prostoru (podle konvence z souřadnice rostou ve směru pohledu pozorovatele hledícího na grafický obsah).
- ***pointsAtX, pointsAtY, pointsAtZ*** - poloha cíle (bodu), na který je reflektor zaměřen.
- ***specularExponent*** - hodnota exponentu, určujícího zaostření reflektoru; výchozí hodnota se rovná 1.
- ***limitingConeAngle*** - úhel omezující osvětlenou oblast je měřen od osy světelného kužele; pokud tuto hodnotu nedefinujeme, nebude nasvícená oblast nijak omezena. (18)

10 Animace

Deklarativní animace umožňují velmi snadné animování grafických objektů bez použití skriptů. Tento způsob se označuje jako "time-based" - umožňuje definovat přesně načasovanou změnu v podstatě jakéhokoli parametru grafiky, počínaje rozměry, polohou a průhledností až třeba po změnu transformačních atributů. To je hlavní rozdíl oproti SWF formátu, kde se časování animací vždy odvíjí od jednotlivých klíčových snímků - anglicky "frame-based". Další důležitý rozdíl je v tom, že jednotlivé deklarativní animace mohou být na sobě zcela nezávislé a mít jinou dobu trvání nebo počet opakování a podobně.

Technologie SMIL má už dnes poměrně silnou podporu v přehrávačích jako QuickTime nebo RealPlayer.

10.1 Odkazy v SVG

Ještě než se ponoříme do animací, povíme si něco málo o tom, jak v SVG vytvářet odkazy. Ty jsou totiž základem interaktivity. Obdoba hyperlinku známého z klasického HTML vypadá takto:

```
<a xlink:href="http://www.w3.org">
  <ellipse cx="2.5" cy="1.5" rx="2" ry="1"  fill="red" />
</a>
```

Všechny grafické objekty uvnitř elementu hyperlinku jsou aktivní - reagují jako odkaz. v uvedeném příkladu je to červená elipsa.

Dalšími možnými atributy jsou xlink:title pro pojmenování odkazu a target pro jméno rámu ve kterém se má otevřít cílový dokument.

SVG má ve slovníku i analogii odkazu na konkrétní prvek ve stránce. Jelikož se v případě SVG jedná o grafický objekt - obrázek, je důležité mít možnost adresovat konkrétní výřez kresby, jinak řečeno pohled, anglicky "view". v zásadě jsou k dispozici dvě možnosti:

1. **kresba.svg#svgView(viewBox**(0,200,1000,1000)) - přesným číselným definováním výřezu grafiky
2. **kresba.svg#nejaky_pohled** - odkazem na speciální prvek view obsahující předdefinovaný pohled na grafický dokument

10.2 Element view

Přehled atributů elementu:

- ***viewTarget*** = (jméno_objektu) - (nepovinný) pokud zadáme identifikátor objektu, přiřazeného ke konkrétnímu pohledu, prohlížeč by jej měl po aktivaci hyperlinku zvýraznit
- ***viewBox*** = (x y šířka výška) - atribut umožní definovat virtuální okno, kterým se uživatel po aktivaci hyperlinku bude na dokument dívat: x, y nastaví levý horní roh; šířka a výška pak určují rozsah virtuálního pohledu
- ***preserveAspectRatio*** = (typ_přizpůsobení_rozměrů) - použitím "viewBox" může dojít i k neproporcionální deformaci pohledu, proto máte k dispozici tento parametr, kterým můžete nastavit chování prohlížeče: "none" umožní deformaci souřadnic a hodnota, například "xMidYMid", přizpůsobí grafiku proporcionálně do pohledového okna a navíc ji vycentruje v obou směrech (toto je výchozí hodnota).
- ***zoomAndPan*** - nastavením na hodnotu "disable" mohou prvky view (taktéž všechny vnější prvky svg) zakázat uživateli manuálně měnit pohled a zvětšení v prohlížeči SVG (výchozí je "magnify" - vše povoleno)

Klíčové pro pochopení je slovíčko **deklarativní** - tvůrce animace přesně definuje (deklaruje), že daný atribut konkrétního objektu bude animován (čili jeho hodnota se bude měnit) v definovaném rozmezí hodnot v konkrétním čase od-do. SVG norma má obrovskou škálu možností, jak to udělat. Velmi flexibilní je i určování času - animace mohou být například startovány nějakou událostí uživatelského rozhraní (třeba "onmouseover") nebo navázány na průběh nějaké jiné SVG animace.

10.3 Vztah SVG vůči SMIL

SVG je jazyk odvozený od W3C normy SMIL ("host language" v terminologii SMIL) a jako takový tedy obsahuje animační prvky v ní definované - kromě nich ale byly definovány i některé další, aby se dosáhlo pokrytí celé oblasti 2D grafiky.

Prvky definované ve specifikaci SMIL:

- ***animate*** - mění hodnoty skalárních atributů v čase
- *set* - zjednodušení prvku animate používané pro změnu (animování) nečíselných atributů typu visibility
- ***animateMotion*** - pohybuje s prvkem podél zvolené animační křivky určující dráhu pohybu
- ***animateColor*** - pro změnu barevných hodnot

Prvky vztahující se speciálně k SVG animacím

- ***animateTransform*** - animuje transformační atributy, velmi silný nástroj - jelikož je možné jednotlivé typy animací kombinovat, lze pohodlně vytvořit libovolné pohybové efekty
- ***mpath*** element - vektorová cesta, kterou lze umístit do prvku animateMotion a definovat tak přesně dráhu pohybu
- ***keyPoints*** - tento attribut je dalším rozšířením elementu animateMotion a umožňuje řídit zrychlení, nejčastější použití je evidentní - vytvoření přirozených pohybových efektů
- ***rotate*** - další atribut pro prvek animateMotion, který řídí směr natočení animovaného objektu (například díváme-li se shora na auto pohybující se po křivce, jeho špička by se samozřejmě měla stále natáčet ve směru jízdy)

10.4 Základní struktura zápisu animačních prvků - atributy

V této části si probereme ty atributy, které naleznou využití ve většině animačních prvků.

Odkaz na animovaný element

Atribut "**xlink:href**" obsahuje URI odkazující na cílový element, jenž musí být v aktuálním SVG fragmentu, aby byla zaručena jeho dostupnost. URI musí ukazovat na *přesně jeden* element. Toto URI lze dokonce vynechat - animován pak bude *rodičovský prvek* přímo nadřazený danému objektu animate.

Určení animovaného atributu

Atribut "**attributeName**" - zde určíte atribut, který byste chtěli v animaci měnit - pro animaci pohybu to může být třeba "x".

Atribut "**attributeType**" - možné hodnoty: "CSS", "XML", "auto". Určení jmenného prostoru XML ve kterém je definován cílový atribut. Výchozí hodnota je "auto" a v naprosté většině případů nebude vůbec nutné se tímto parametrem zabývat.

Určení rozsahu hodnot

Následující tři atributy umožňují nastavit rozmezí, ve kterém se v průběhu animování budou hodnoty měnit. (V prvku set se uplatní jen posledně jmenovaný.)

- ***from*** - počáteční hodnota animovaného atributu.
- ***by*** - změna hodnoty vůči počátečnímu stavu. Pokud nastavíme "by", nebudeme už nastavovat konečnou hodnotu "to" a naopak - je-li určena absolutní konečná hodnota, nebudeme nastavovat "by".
- ***to*** - konečná hodnota animace.

Zachování výsledku animace

Atributem "**fill**" můžete určit, zda pracovní výsledek animace zůstane zachován (freeze) i po jejím skončení nebo bude odstraněn (remove - výchozí hodnota).

Aditivnost a kumulativnost animovaných hodnot

Velmi často je při animaci užitečné mít možnost měnit nějakou hodnotu (polohu, natočení) relativně k počátečnímu, klidovému stavu. Představme si například "roztřesení" animovaného objektu okolo jeho klidové polohy - užijete-li následujících atributů, nebudete se už muset vůbec zajímat o jeho absolutní polohu. (Pro animace polohy se samozřejmě nabízí ještě alternativní řešení této úlohy pomocí změny souřadnicového systému, ale to nyní není důležité.)

Atribut "**additive**" určí, zda aktuální animovaná hodnota nahradí klidovou hodnotu atributu (replace - výchozí hodnota) nebo se k ní bude přičítat (sum).

Atribut "**accumulate**" - kumulativnost se uplatní v případě animačních cyklů, které proběhnou více než jednou. Funguje to přesně tak, jak byste čekali - jestliže ji nastavíte na hodnotu "sum", pak se k aktuální hodnotě animace přičítá hodnota, kterou měl animovaný atribut na konci předcházejícího cyklu.

Příklad použití atributů

```
<rect width="20px" ...>
  <animate attributeName="width"
           from="0px" to="10px" dur="10s"
           additive="sum" accumulate="sum"
           repeatCount="5"/>
</rect>
```

10.5 Základní struktura zápisu animačních prvků - elementy

Element animate

Zcela obecný prvek, použitelný k animaci všech číselných hodnot. v dalších částech poznáte specializovanější elementy, určené třeba jen a pouze k animaci pohybu...

Pomocí jednoho animačního prvku lze měnit v čase právě jeden atribut. Ne všechny atributy a ne všechny elementy mohou být zapojeny do deklarativních animací. Nebojte se ale, většinou bude vše velmi intuitivní a případné nejasnosti si můžete ověřit v tabulkových přehledech normy SVG.

Element set

Tento prvek provádí přesně to, co naznačuje jeho anglický význam - nastaví daný atribut na danou hodnotu v přesně definovaný časový úsek. Využijete ho většinou při operacích s dvoustavovými proměnnými, například typu viditelnost (skryj či ukaž nějaký prvek) nebo změna textu. Tento animační prvek logicky nemůže pracovat aditivně ani kumulativně.

Atribut "to" určí hodnotu, která bude nastavena po dobu trvání animace.

Element animateColor

Atributy "from", "by" a "to" v tomto případě samozřejmě specifikují hodnoty barev, přičemž je několik povolených způsobů zápisu - analogicky, tak jak je zapisujete když definujete barvy výplní a tahů. Do hry vstupuje též atribut "color-interpolation" popsaný již dříve v části o vyplňování.

Element animateTransform - dynamické 2D transformace

Tento prvek umožňuje animovat transformační atributy normy SVG. a ty dovolují elegantně uskutečnit naprosto libovolné polohové transformace, změnu měřítka, zkosení...

Atributy elementu animateTransform:

- ***attributeName*** - tento atribut bude vždy nastaven takto: attributeName="transform"
- ***type*** - určení typu transformace
- ***from***, *by*, *to* - protože většina transformačních operací má více než jeden parametr, budou v tomto případě obsahem těchto atributů všechny požadované hodnoty. Například rotace s počátečním úhlem 30 stupňů a středem otáčení v bodě [25,25] se zapíše takto:

```
<animateTransform attributeName="transform" type="rotate"
   from="30,25,25" to="90,65,65"
   begin="click" dur="9s"
    additive="sum" fill="freeze" />
```

Typy transformací (hodnoty v hranatých závorkách jsou nepovinné, volitelné podle typu požadované transformace):

- "***translate***" - posun_x [,posun_y]
- "***scale***" - měřítko_x [,měřítko_y] - zapíše-li se zde pouze jedna hodnota, bude použita pro změnu měřítka v obou souřadnicových osách, jinak bude násobení pro každou osu rozdílné
- "***rotate***" - úhel [střed_x střed_y] - pozor, neuvede-li se na tomto místě střed otáčení, bude se dotyčný objekt otáčet nikoli okolo své osy, ale okolo bodu se souřadnicemi (0,0)
- "***skewX***", "***skewY***" - úhel zkosení

Příklad použití animačních elementů

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

<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     viewBox="0 0 480 240" width="240px"
     version="1.1" baseProfile="basic">
<title>9.7.2 Animace</title>
<desc>
  Priklad animuje transformaci vyplnoveho vzoru
  a ukazuje jak snadno muzete v SVG vyrobit
  i relativne slozite graficke hracky.
</desc>
  <!-- definice textury slozene ze 4 ctvercu -->
  <!-- navic zdeformovanych transformacnim atr. -->
  <pattern patternUnits="userSpaceOnUse" id="Pat3"
           x="0" y="0" width="20" height="20"
           patternTransform="translate(25 215) scale(2)
skewX(45)">
    <rect x="0" y="0" width="10" height="10" fill="red"/>
    <rect x="10" y="0" width="10" height="10" fill="green"/>
    <rect x="0" y="10" width="10" height="10" fill="blue"/>
    <rect x="10" y="10" width="10" height="10" fill="yellow"/>
    <!-- toto vytvari dojem nekonecneho beziciho pasu -->
    <animateTransform attributeName="patternTransform"
      type="translate" from="0,0" to="40,0" dur="1s"
      fill="freeze" additive="sum" repeatCount="indefinite" />
  </pattern>
  <!-- textura se aplikuje na objekty -->
  <rect x="20" y="30" width="440" height="50" fill="url(#Pat3)"/>
  <circle cx="400" cy="120" r="40" fill="url(#Pat3)" />
  <rect x="20" y="100" width="100" height="50" fill="url(#Pat3)">
    <!-- vytvari efekt smrstovani a natahovani -->
    <animateTransform attributeName="transform"
      type="scale" values="1,1;3,1;1,1" keyTimes="0 85% 1"
dur="8s"
      fill="freeze" additive="sum" repeatCount="indefinite" />
  </rect>
  <text font-family="Helvetica" font-size="32" x="20" y="220">
    Vzorky jsou animovany.
  </text>
<!-- obrys platna -->
<rect x="1" y="1" width="478" height="238"
      fill="none" stroke-width="2" stroke="blue" />
</svg>
```

(34)

10.6 Animace (časování)

S této části se budeme věnovat řízení spouštění a opakování animací, které jsou jistě jednou z nejpřitažlivějších oblastí 2D grafiky. Animační prvky SVG-SMIL mají schopnost reagovat na většinu událostí z uživatelského rozhraní počítače (mobilu) jen s použitím vlastních prostředků, bez dodatečného skriptování. Ale pokud umíte JavaScript, můžete jej samozřejmě použít též.

Atributy pro řízení časového průběhu animací

Možností, jak řídit časování animací, je v SVG tolik a jsou propracované. Jistě stojí za to se jim konkrétně věnovat.

Atribut begin

Atribut "begin" definuje počátek animace (norma to někdy nazývá též "aktivace animačního elementu"). Může obsahovat celý seznam možných hodnot navzájem oddělených středníky, z čehož plyne, že jedna animace může být spouštěna ne jednou, nýbrž několika různými akcemi a událostmi:

- ***offset-value*** - čas vyjádřený v sekundách, zjednodušeně můžeme říci, že "0s" je čas načtení SVG dokumentu (pokud vás zajímají detaily, najdete je ve specifikaci SMIL)
- ***syncbase-value*** - tento typ hodnot umožňuje synchronizovat animaci s počátkem či koncem jiné animace, zapíše se jako "id" synchronní animace, následované tečkou a klíčových slůvkem "begin" či "end", například begin="id_jine_animace.end", navíc je možné přidat časový posun, například begin="id_jine_animace.start+3s" (posun obdobně funguje i v následujících třech bodech)
- ***event-value*** - tento typ je velmi podobný předcházejícímu a umožní reagovat na všechny možné události (events) - ano, z HTML důvěrné známé hodnoty typu "click" nebo "mouseover" plus nějaké SVG specialitky - kompletní výpis najdete v normě SVG, příklad: begin="id_jine_animace.repeatEvent"
- ***repeat-value*** - analogický k dvěma předcházejícím, ale umožní reakci na určitý počet opakovaných spuštění nějaké cyklické animace, například begin="id_cyklicke_animace.repeat(5)"
- ***accessKey-value*** - dává možnost reagovat na stisk klávesy, například begin="id_animace.accessKey('x')"
- ***wallclock-sync-value*** - spuštění v přesně definovaný reálný čas, jako třeba 12.00 hod. v poledne a podobně (viz ISO8601)
- ***indefinite*** - zajímavé postavení má tato hodnota, která, je-li uvedena v seznamu startovacích hodnot, umožní spouštět danou animaci buď voláním funkce "beginElement()" anebo klepnutím na hyperlink zacílený na daný animační prvek

Atribut dur

Atribut "dur" určuje dobu trvání animace (například `12min`). Hodnota "indefinite" znamená nekonečně dlouho.

Atribut end

Atribut "end" určuje čas ukončení animace. Způsoby definování jsou v podstatě shodné s těmi pro startování, jen funkce JavaScriptu ukončující animaci se pochopitelně jmenuje "endElement()".

Atribut restart

Atribut "restart" určuje, zda je animaci možno "restartovat", tedy spustit znovu od začátku. Možné hodnoty jsou:

- ***always*** - restart je možný kdykoli (výchozí hodnota)
- ***whenNotActive*** - restartovat lze, pokud animace není aktivní (neběží)
- ***never*** - restart není možný

Atribut repeatCount

Atribut "repeatCount" určuje počet opakování animace, hodnota "indefinite" značí nekonečno. zajímavé přitom je, že počet opakování může být desetinné číslo a poslední cyklus tudíž nemusí proběhnout celý!

Atribut repeatDur

Atribut "repeatDur" určuje dobu, po kterou se má základní cyklus animace opakovat. Hodnota "indefinite" opět značí nekonečně dlouho.

Příklady zápisu časových údajů podle SMIL

Není-li u hodnoty udávající délku časového intervalu jednotka, prohlížeč údaj považuje za sekundy. (33)

plný tvar:
02:30:03
50:00:10.25 = 50h, 10s a 250ms

zkrácený tvar:
00:10.5
02:33 = 0h, 2m a 33s

délka intervalu:
3.2h
45min
30s
5ms
12.467 = 12s a 467ms

10.7 Ovládání dynamiky změny hodnot

Dynamika - způsob jak se mění animované hodnoty. i když se velmi často vůbec nebudete muset tímto problémem zaobírat, později, zvláště při modelování realistického pohybu, dojdete k potřebě jemnějšího ovládání rychlosti a zrychlení objektu.

Atribut **calcMode** - nepovinný atribut určující způsob interpolace hodnot, možné hodnoty jsou:

- ***discrete*** - skoková změna hodnoty mezi definovanými klíčovými body animace (vyzkoušejte si názornou demonstraci, zdroj: Adobe).
- ***linear*** - aktuální hodnota je v každém okamžiku určena jednoduchou lineární interpolací mezi dvěma definovanými klíčovými body animace (výchozí hodnota).
- ***paced*** - konstantní krok (rychlost) po celou dobu trvání animace. Způsob jen zdánlivě shodný s předcházejícím bodem se uplatní pouze pro případy, kdy lze mezi klíčovými body měřit vzdálenost. Pokud je nastaven typ "paced", budou ignorovány atributy keyTimes a keySplines. Tato metoda je výchozí pro prvky animateMotion.
- ***spline*** - časový průběh hodnot je definován pomocí kubické Bézierovy křivky.

Atribut **values** - seznam hodnot, kterých bude animace postupně nabývat, přičemž jednotlivé animační kroky jsou odděleny středníky. Pokud je tento seznam použit, způsobí ignorování atributů from, to a by - v tomto případě rovněž velice doporučuji přebytečné parametry vůbec neuvádět, aby nedocházelo k nějakým přehmatům.

Atribut **keyTimes** - seznam časů oddělených středníky umožňuje měnit rychlost změny v průběhu animace. Atribut musí být vždy použit ve spojení s hodnotami values, přičemž každá hodnota ze seznamu values má přiřazen svůj klíčový čas uvedený jako desetinné číslo v rozsahu od 0 (počátek animace) do 1 (konec). Atribut je ignorován, pokud je současně nastaven calcMode="paced".

Atribut **keySplines** - jejich funkce je doplňková k atributu keyTimes (proto musí být nastaveny oba seznamy současně), s jediným rozdílem, že v seznamu nejsou klíčové časy, nýbrž sady čtyř řídicích bodů x1 y1 x2 y2 v rozsahu 0 až 1, které pomocí kubických Bézierových křivek určují dynamiku změny hodnot mezi dvěma klíčovými časy - proto také tento seznam bude vždy o jednu sadu hodnot kratší než oba předcházející. Pokud není současně nastaven calcMode="spline", je atribut ignorován.

Pro pochopení funkce **keySplines** je třeba si představit souřadnici x jako časovou osu pro daný úsek animace a y pak jako osu pro výslednou hodnotu animace, přitom y=0 odpovídá počáteční hodnotě animované hodnoty a y=1 je konečná hodnota.

Příklad:

```
<animate dur="5s" attributeName="cy"
      values="0;10" keyTimes="0;1"
      calcMode="spline" keySplines="0 0 1 1" />
<animate dur="5s" attributeName="cy"
      values="0;10" keyTimes="0;1"
      calcMode="spline" keySplines=".5 0 .5 1" />
<animate dur="5s" attributeName="cy"
      values="0;10" keyTimes="0;1"
      calcMode="spline" keySplines="0 .75 .25 1" />
<animate dur="5s" attributeName="cy"
      values="0;10" keyTimes="0;1"
      calcMode="spline" keySplines="1 0 .25 .25" />
```

Element AnimateMotion

Pohyb po předdefinované dvourozměrné křivce.

Atributy:

- ***calcMode*** - možné hodnoty: discrete, linear, paced, spline - popis viz výše.
- ***path*** - prvek obsahuje data pro vektorovou cestu určující pohyb animovaného objektu, "motion path" se zapisuje zcela stejně jak již dříve vysvětlený atribut "d" v prvku path.
- ***keyPoints*** - obsahuje středníky oddělený seznam desetinných čísel (klíčových bodů) v rozsahu od 0 do 1 a určuje, jak daleko na cestě se má pohybující se objekt nacházet v příslušném čase, uloženém v seznamu klíčových bodů animace keyTimes, viz výše.
- ***rotate*** - určuje způsob natáčení objektu ve vztahu k cestě po které se pohybuje:
    - **auto** - objekt se natáčí ve směru tečny pohybové křivky (tedy v již zmíněném příkladu s autem animovaným na silnici v pohledu shora se bude automobil správně otáčet přídí ve směru jízdy)
    - **auto-reverse** - stejně jako "auto", ale objekt je navíc otočen o 180 stupňů

- "**úhlová_hodnota**" - pokud zadáte úhel měřený od osy x, animovaný objekt bude stále zachovávat toto natočení; výchozí hodnota je 0

Směr pohybu lze zadávat i jednodušeji, pomocí obligátních atributů "from", "by", "to", jejich hodnoty jsou v tomto případě tvořeny páry hodnot oddělených mezerou nebo čárkou, například from="33,15". Další možnost pomocí již známého atributu "values" může vypadat nějak takto values="10,20;30,20;30,40" nebo values="10mm,20mm;30mm,20mm;30mm,40mm".

Alternativní definice pohybové křivky - mpath

Tento prvek musí být vnořen uvnitř elementu animateMotion.

Atributy:

- ***xlink:href*** - URI odkaz na prvek path, pokud je tento odkaz definován, má přednost před atributem path v rodičovském elementu animateMotion.

10.8 Vzdálenost podél cesty

Nejenom při pohybu po křivce, ale i při některých dalších grafických operacích (jako třeba umísťování textu na křivku) je potřeba vypočítat délku měřenou podél konkrétní křivky. Matematická věda to samozřejmě umí, ale jelikož se v žádném případě nejedná o jednoduché výpočty, norma nenutí prohlížeče implementovat absolutně přesné řešení.

Místo toho je nabídnuta alternativní možnost použít dodatečný atribut "pathLength", ve kterém autor prohlížeči sdělí skutečnou délku dotyčné křivky, která pak bude zpětně prohlížečem použita pro umísťování objektů podél křivky. Kreslicí příkazy "moveto" mají pro účely výpočtu délky křivky délku 0! (32)

11 SVG versus Flash

Macromedia SWF formát byl již při vývoji SVG relativně rozšířený a tak pracovní skupina SVG přirozeně vycházela také ze zkušeností vývojářů Flashe. SVG norma nabízí všechny kladné vlastnosti Flash technologie. Výhodou je lepší práce s písmy a dokonalejší provázanost se zbytkem internetu, včetně přirozeného vytváření hyperlinků a skriptovatelnosti. V této části se podíváme na rozdílnost obou vektorových formátů. Nebudeme zmiňovat stejné vlastnosti.

Hodnocení SVG

- ++ ***otevřenost*** - umožňuje v podstatě komukoli vytváření a úpravu grafiky - v případě nutnosti postačí textový editor.
- ++ **XML základy** - stoprocentně a zcela systémově propojitelný s celým internetovým světem, v budoucnosti, díky tomu získáte například velmi elegantní možnost vkládání SVG kódu přímo do XHTML dokumentu
- ++ **XML struktura** a s ní spojené techniky jako skriptovatelnost a manipulace s grafikou přes DOM, možnost nasazení stylů CSS, model událostí společný všem XML jazykům, potažmo tedy shodný s HTML, to vše jsou věci, které weboví vývojáři důvěrně znají a mohou tedy tyto znalosti okamžitě použít i pro SVG.
- ++ **snadná možnost automatizovaného generování SVG na serveru** pro vizualizaci databázových dat.
- ++ **Inkrementální načítání**. Podobně, jako to dělají některé prohlížeče HTML. Při načítání z pomalé sítě lze ihned zobrazovat načtené elementy a nečekat na načtení celého dokumentu (každý z nás jistě zažil při načítání mnohých "flashnutých" stránek neskutečně otravné čekací lhůty).
- **-- SVG je malinko handicapován zatím nedostatečným rozšířením plug-inů do internetových prohlížečů**, ale tato ztráta se rychle zmenšuje.
- ++ Na druhou stranu má SVG díky své otevřenosti už dnes převahu na mobilních telefonech a komunikátorech, kde již existují implementace normy SVG Tiny a SVG Basic. Norma *SVG Tiny* byla dokonce zvolena jako povinný základ Multimedia Message Service (MMS) pro novou generaci mobilů podle normy 3GPP!
- -- Díky tomu, že SVG je mladší než SWF, stále čeká na podobně vyspělé kreativní prostředí, kterým je Macromedia Flash MX

Hodnocení Flash technologie

- ++ **Zásadní výhodou Flashe je Flash**. Toto, na první pohled paradoxní tvrzení, se objeví v jiném světle, uvědomíme-li si, že Flash je především aplikace pro vytváření animací a interaktivních audio-vizuálních prezentací. Program se současným názvem Flash MX má za sebou obrovskou historii, začínající na počítačích Apple už v době, kdy uživatelé PC teprve zjišťovali, že myši běhají nejenom ve sklepě. a tyto zkušenosti jsou na něm vidět... (Historická poznámka: Roku 1984 byla založena společnost Macromind s cílem dát interaktivitu, multimédia a animace do rukou neprogramátorům, v roce 1987 uživatelé dostávají do rukou první dostatečně výkonný nástroj - první barevný Macintosh, v roce 1988 přichází další významný milník - aplikace Macromind Director 1 spolu s nástupem CD mechanik, 1991 sloučení Macromind-Paracomp, 1992 další sloučení s Authorware a vznik Macromedie.)
- ++ **prohlížeč SWF formátu je starší** a je rozšířenější a lépe optimalizovaný na rychlost.
- ++ Díky tomu, že Flash plug-in se vyvíjí v těsné vazbě na grafickou aplikaci Flash MX, může se grafik vždy spolehnout na dokonalou funkčnost na všech platformách podporovaných Macromedií - tedy pokud zrovna uživatel není s instalací opožděný o jednu vývojovou verzi plug-inu.
- -- **Formát je uzavřený a autoři jsou odkázáni na aplikace**, které export SWF podporují. Můžete udělat právě jen to, co vám daný software umožní.
- -- **chcete-li pracovat s formátem SWF, musíte se nejdříve kompletně od základu naučit celou specializovanou Flash technologii.** (40)

12 Závěr

Není třeba dlouze polemizovat o tom, zda-li zpracování grafiky pomocí SVG bude mít v budoucnosti reálné využití. Věnovala jsem se podpoře SVG ve velké míře právě proto, že přichází doba nativní podpory v nejpoužívanějších prohlížečích. Velká výhoda spočívá v otevřenosti kódu, která umožňuje modifikace prakticky v každém textovém editoru, nehledě na velkou škálu profesionálních aplikací jak pro programátory, tak pro začínající grafiky. Zlí jazykové tvrdí, že SVG spíše odrazuje problematikou zobrazení než jeho výhodami. Kdo ale zkusil jednou možnosti efektů a krásně hladkých hran v jakékoliv velikosti vykreslení, brzy tomuto formátu přijde na chuť. Znalci xml si mohou libovat v podobnosti syntaxe a v podstatě mohou neprodleně zkoušet krásy SVG. Jednou z částí mé práce je také rozsáhlá galerie praktických příkladů, na kterých můžete začít hned zkoušet své první kroky s technologií SVG. Najdete ji na adrese: http://pepers-terez-web.ic.cz/bak/ nebo na přiloženém CD. Závěrem tedy volám: „Nebojte se, časy SVG právě přicházejí. Využijte rychlosti vykreslení, formátujte, transformujete, animujte! Těšte se z krás vektorové grafiky při tvorbě webových prezentací.“

Použité zdroje:

1) Hejral, Martin. Průvodce SVG. Interval.cz[online]. 15. 4. 2003. [cit. 2009-01-27]. URL: <http://interval.cz/clanky/pruvodce-svg/>

2) Hejral, Martin. Kurz SVG - struktura dokumentu, zobrazovací a vykreslovací model. Interval.cz [online]. 7. 7. 2004. [cit. 2009-01-27]. URL: <http://interval.cz/clanky/kurz-svg-struktura-dokumentu-zobrazovaci-a-vykreslovaci-model/>

3) Hejral, Martin. Kurz svg – grafická primitiva. Interval.cz [online]. 14. 7. 2004. [cit. 2009-01-27]. URL: <http://interval.cz/clanky/kurz-svg-graficka-primitiva/>

4) Hejral, Martin. Průvodce svg – grafické editory. Interval.cz [online]. 10. 9. 2003. [cit. 2009-01-27]. URL: <http://interval.cz/clanky/ pruvodce-svg-graficke-editory />

5) Tišnovský, Pavel. Root.cz[online]. 2. 8. 2007. [cit. 2009-01-27]. URL: <http://www.root.cz/clanky/vektorovy-graficky-format-svg/>

6) Scalable Vector Graphics. Wikipedie [online]. 14. 11. 2008. [cit. 2009-01-27]. URL: <http://cs.wikipedia.org/wiki/Scalable_Vector_Graphics>

7) Basic Shapes with SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL: <http://www.svgbasics.com/shapes.html>

8) Rotating Shapes with a Transform in SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:< http://www.svgbasics.com/rotate.html>

9) Simple Lines with SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/lines.html>

10) Line Markers with SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/markers.html>

11) Describing Quadratic Bézier Curves in SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/curves.html>

12) Specifying a Font for Text with SVG. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/using_fonts.html>

13) Decorating Text with SVG - Italic, Subscript and Superscript. SVGBasics[online]. 2004.[cit. 2009-03-13]. URL:<http://www.svgbasics.com/font_effects_italic.html>

14) Decorating Text with SVG - Bold, Underline, Overline and Strikethrough. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/font_effects_bold.html>

15) Kosík, Zbyněk. SVG má zelenou. Zive.cz[online]. 11. 9. 2001. [cit. 2009-03-31]. URL:<http://www.zive.cz/Clanky/SVG-ma-zelenou/sc-3-a-102835/default.aspx/>

16) Adding Gloss - Using Filters for Lighting. SVGBasics[online]. 2004. [cit. 2009-03-13]. URL:<http://www.svgbasics.com/filters2.html>

17) feOffset - a Filter for Drop Shadows. SVGBasics[online]. 2004.[cit. 2009-03-13]. URL:<http://www.svgbasics.com/filters3.html>

18) Hejral, Martin. Kurz SVG - bitmapové efekty (definice W3C, parametry osvětlení). Interval.cz[online]. 9. 11. 2005. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-bitmapove-efekty-definice-w3c-parametry-osvetleni/>

19) Hejral, Martin. Kurz SVG - grafická primitiva .Interval.cz[online]. 14. 7. 2004. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-graficka-primitiva/>

20) Hejral, Martin. Kurz SVG - vyplňování 1.Interval.cz[online]. 28. 7. 2004. [cit. 2009-03-13]. URL:< http://interval.cz/clanky/kurz-svg-vyplnovani-1/>

21) Hejral, Martin. Kurz SVG - vyplňování 2.Interval.cz[online]. 3. 8. 2004. [cit. 2009-03-13]. URL:< http://interval.cz/clanky/kurz-svg-vyplnovani-2/>

22) Hejral, Martin. Kurz SVG - ořezávání a maskování. Interval.cz[online]. 11. 8. 2004. [cit. 2009-03-13]. URL:< http://interval.cz/clanky/kurz-svg-orezavani-a-maskovani/>

23) Hejral, Martin. Kurz SVG - bitmapové efekty (úvod a galerie). Interval.cz[online]. 20. 10. 2005. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-bitmapove-efekty-uvod-a-galerie/>

24) Hejral, Martin. Kurz SVG - bitmapové efekty (elementární grafické filtry 1.). Interval.cz[online]. 1. 2. 2006. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-bitmapove-efekty-elementarni-graficke-filtry-1/>

25) Hejral, Martin. Kurz SVG - bitmapové efekty (elementární grafické filtry 2.). Interval.cz[online]. 8. 2. 2006. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-bitmapove-efekty-elementarni-graficke-filtry-2/>

26) Macich, Jiří. SVG bude možná podporovat až Internet Explorer 9. Lupa.cz[online]. 23. 1. 2009. [cit. 2009-04-16]. URL:<http://www.lupa.cz/zpravicky/svg-bude-mozna-podporovat-az-internet-explorer-9/>

27) Mikula, Jan. První zmínky o Internet Exploreru 9: Rychlejší Java Script, podpora SVG a zaoblené rohy. Prohlizece.info[online]. 23.01.2009. [cit. 2009-04-16].

URL:<http://prohlizece.info/clanky/prvni-zminky-o-internet-exploreru-9-rychlejsi-java-script-podpora-svg-a-zaoblene-rohy/>

28) Krčmář, Petr. Firefox 1.5: dospělý prohlížeč.Root.cz[online]. 30. 11. 2005. [cit. 2009-04-16]. URL:<http://www.root.cz/clanky/firefox-1-5-dospely-prohlizec/>

29) Amaya. Prohlizece.info[online]. [cit. 2009-04-16]. URL:<http://prohlizece.info/amaya/>

30) SVG v kartografii.geoinformatics.fsv.cvut.cz [online]. [cit. 2009-04-16]. URL:<http://geoinformatics.fsv.cvut.cz/gwiki/SVG_v_kartografii>

31) Hejral, Martin. Kurz SVG - animace (praktické ukázky pro pokročilé). Interval.cz [online]. 10. 11. 2004.[cit. 2009-04-16]. URL:<http://interval.cz/clanky/kurz-svg-animace-prakticke-ukazky-pro-pokrocile/>

32) Hejral, Martin. Kurz SVG - animace (dynamika změny hodnot). Interval.cz [online]. 12. 10. 2004. [cit. 2009-04-16]. URL:<http://interval.cz/clanky/kurz-svg-animace-dynamika-zmeny-hodnot/>

33) Hejral, Martin. Kurz SVG - animace (časování). Interval.cz [online]. 15. 9. 2004. [cit. 2009-04-16]. URL:<http://interval.cz/clanky/kurz-svg-animace-casovani/>

34) Hejral, Martin. Kurz SVG - není to Flash a přece se točí!.Interval.cz [online]. 2. 9.2004. [cit. 2009-04-16]. URL:<http://interval.cz/clanky/kurz-svg-neni-to-flash-a-prece-se-toci/>

35) Hejral, Martin. Průvodce SVG - Inkscape a Sketsa, SVG editory pro každého .Interval.cz [online]. 28. 7. 2006. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-inkscape-a-sketsa-svg-editory-pro-kazdeho/>

36) Hejral, Martin. Průvodce SVG - Scalable Vector Graphics v polovině roku 2006. Interval.cz [online]. 13. 7. 2006. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-scalable-vector-graphics-v-polovine-roku-2006/>

37) Hejral, Martin. Průvodce SVG - XStream RapidSVG a Ikivo Animator. Interval.cz [online]. 2. 2. 2005. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-xstream-rapidsvg-a-ikivo-animator/>

38) Hejral, Martin. Průvodce SVG - XStudio, Inkscape a další SVG editory. Interval.cz [online]. 16. 3. 2005. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-xstudio-inkscape-a-dalsi-svg-editory/>

39) Hejral, Martin. Průvodce SVG - SVGDeveloper a XFY. Interval.cz [online]. 8. 8. 2006. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-svgdeveloper-a-xfy/>

40) Hejral, Martin. Průvodce SVG - SVG versus Flash. Interval.cz [online]. 28. 4. 2003. [cit. 2009-04-18]. URL:<http://interval.cz/clanky/pruvodce-svg-svg-versus-flash/>

41) Hejral, Martin. Kurz SVG – text. Interval.cz[online]. 20. 7. 2004. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/kurz-svg-text/>

42) Hejral, Martin. Průvodce SVG - hlášení o stavu vývoje. Interval.cz[online]. 23. 4. 2003. [cit. 2009-03-13]. URL:<http://interval.cz/clanky/pruvodce-svg-hlaseni-o-stavu-vyvoje/>