INSTYTUT INŻYNIERII I GOSPODARKI WODNEJ

POLITECHNIKA KRAKOWSKA im. TADEUSZA KOŚCIUSZKI

Andrzej Jurzec

WYKORZYSTANIE FORMATU SVG W SYSTEMACH INFORMACJI PRZESTRZENNEJ

praca magisterska

studia dzienne

kierunek studiów: **informatyka**

specjalność: **informatyka stosowana w inżynierii środowiska**

promotor: **dr inż. Robert Szczepanek**

nr pracy:

KRAKÓW 2009

ul. Warszawska 24, 31-155 Kraków tel/fax (+48 12) 628 20 41 e-mail: sekretariat@iigw.pl internet: www.iigw.pl

Składam serdeczne podziękowania mojej żonie Ani.

Spis Treści

Wstęp

Wykorzystanie grafiki wektorowej jest nieodzowne w pracach inżynieryjno–projektowych. Format grafiki wektorowej niesie ze sobą liczne korzyści, ale też problemy.

O ile w przypadku grafiki rastrowej istnieje wiele formatów (BMP, JPEG, GIF), które potrafią obsłużyć wszystkie programy zajmujące się wyświetlaniem lub edycją tejże grafiki, tak w przypadku grafiki wektorowej takiego uniwersalnego formatu, który by zdobył szeroką popularność, nie ma. Wynika to poniekąd z faktu, że grafika wektorowa jest skomplikowana, trudna do opisania i nie ma sposobu jej zapisu, który można by było uznać za najlepszy. Z kolei firmy zajmujące się produkcją oprogramowania wolą mieć swój format zapisu, który jest przez nich pilnie strzeżony, co wymusza na klientach korzystanie z produktów danej firmy.

Celem mojej pracy jest przybliżenie formatu grafiki wektorowej SVG (Scalable Vector Graphics). Pokazanie, że posiada on możliwości, by stać się uniwersalnym formatem grafiki wektorowej, niezależnym od systemu operacyjnego ani od platformy sprzętowej.

W rozdziale pierwszym przedstawiam informacje wstępne dotyczące grafiki rastrowej i wektorowej, języka XML, na którym zbudowany jest format SVG, oraz informacje dotyczące samego formatu SVG.

W rozdziale drugim przybliżam najważniejszą część specyfikacji SVG, co pozwala poznać szerokie możliwości tego formatu.

Natomiast w rozdziale trzecim zamieściłem przykłady systemów informacji przestrzennej, w których jest używany format SVG, oraz programów, które nie są związane z systemami GIS, jednak dzięki nim można edytować lub wyświetlać pliki zapisane w formacie SVG.

1. Wprowadzenie do SVG

1.1. Grafika rastrowa

Obraz grafiki rastrowej jest plikiem reprezentującym prostokątną siatkę pikseli. Kolor każdego piksela jest definiowany osobno. Jakość obrazka rastrowego jest określana przez całkowitą liczbę pikseli (wielkość obrazu) oraz ilość informacji przechowywanych w każdym pikselu (głębia koloru). Najczęściej spotykane głębie koloru to[1]:

- **1-bit** – 2 kolory;
- **8-bit** – umożliwia wyświetlenie 2^8, czyli 256 kolorów. Zielony i czerwony kodowane są na 3 bitach każdy, a niebieski na pozostałych dwóch.
- **16-bit High Color** – umożliwia wyświetlenie 2^{16}, czyli 65536 kolorów. Czerwony i niebieski kodowane są na 5 bitach, co daje po 32 odcienie każdego z nich. Zielony kodowany jest na 6 bitach, co daje 64 możliwe odcienie. Zielony kodowany jest na 1 bicie więcej, ponieważ oko ludzkie rozpoznaje więcej odcieni koloru zielonego;
- **24-bit True Color** – umożliwia wyświetlenie 2^{24}, czyli 16777216 kolorów. Koduje każdy z kolorów RGB na 8 bitach, co daje 256 odcieni każdego koloru;
- **> 24-bit** – liczba kolorów jest taka sama jak w zapisie 24 bit, przy czym dodatkowe bity wykorzystywane są na dane niedotyczące koloru takie jak: przezroczystość alfa, odległość "z", chropowatość, czy inne informacje ułatwiające przekształcenia obrazu i wyeliminowanie niepożądanych skutków tych przekształceń.

Ponieważ obrazy rastrowe zajmują stosunkowo dużo miejsca (obraz o rozmiarze 800x600 pikseli o głębi koloru 24bit zajmuje prawie 1,4 MB), często stosuje się technikę kompresji danych celem zmniejszenia wielkości zajmowanego miejsca. Niektóre techniki zmieniają (zmniejszają, usuwają) pewne informacje, aby uzyskać mniejszy plik. Niestety

[1] *http://en.wikipedia.org/wiki/Color_depth*

nie są to bezstratne metody kompresji. Przykładami takich kompresji są JPEG, GIF (obcinanie do głębi 8bit).

Zalety grafiki rastrowej:

- dobrze oddaje półcienie;
- łagodne przejścia między kolorami;
- subtelne cieniowanie obrazów ciągłotonowych;
- mnogość narzędzi do wyświetlania i edycji;
- popularne formaty zapisu.

Wady grafiki rastrowej:

- skalowanie obrazu powoduje utratę jego jakości;
- źle wygląda przy dużych powiększeniach;
- stosunkowo duże rozmiary.

Zastosowanie grafiki rastrowej:

- zapisywanie zdjęć i realistycznych obrazów – rysunków, grafik;
- efekt końcowy skanowania obrazów analogowych;
- przechowywanie zawartości okna ekranowego aplikacji;
- tworzenie obrazów o skomplikowanych kolorach, przejściach tonalnych, gradacjach barw itp.;
- obróbka zdjęć np. retuszowanie;
- tworzenie grafiki ekranowej dla aplikacji multimedialnych;

1.2. Grafika wektorowa

„Obraz wektorowy opisany jest za pomocą figur geometrycznych, tzw. obiektów, które zbudowane są z podstawowych elementów nazywanych prymitywami, czyli prostych figur geometrycznych takich jak odcinki, krzywe, okręgi, wielokąty. Obiekty maja swoje współrzędne (umiejscowienie, wysokość, szerokość itp) oraz atrybuty, mówiące np. o grubości i kolorze linii, kolorze wypełnienia figury lub wypełnieniu niejednolitym jak wypełnienie gradientem albo wzorem, stopniu przezroczystości”[2].

[2] *http://www.ikkiz.pl/*

Zalety:

- stała jakość obrazu niezależnie od tego w jakiej skali zostanie on wyświetlony;
- możliwość edycji pojedynczych obiektów niezależnie od pozostałych;
- stosunkowo niewielkie rozmiary plików (dla obrazów nie fotorealistycznych);
- możliwość opisu przestrzeni 3D (nie wszystkie formaty).

Wady:

- duża złożoność obliczeniowa dla obrazów fotorealistycznych;
- brak uniwersalnego formatu zapisu.

Zastosowanie:

- wykresy;
- rysunki techniczne;
- prezentacja danych i modelowanie;
- prezentacja tekstu;
- zapis dokumentów nie przeznaczonych do dalszej edycji (PDF);
- oprogramowanie do wizualizacji danych (pakiety statyczne, arkusze kalkulacyjne, programy do kreślenia wykresów);
- graficzne środowiska wspomagania projektowania (CAD);
- systemy modelowania rzeczywistości wirtualnej;
- wizytówki, emblematy, znaki firmowe;
- symulacja dla wizualizacji naukowej;
- animacje i gry komputerowe;

1.3. XML

1.3.1. Definicja XML

XML (eXtensible Markup Language, czyli rozszerzalny język znaczników), zaprojektowany został przez World Wide Web Consortium – organizację która stworzyła takie standardy jak HTML, CSS, PNG i inne. XML nie jest językiem do tworzenia dokumentów, lecz jest to rama składniowa do tworzenia języków dla specyficznych zastosowań. Dopiero one mogą służyć do budowania dokumentów. XML może posłużyć

np. do stworzenia wewnętrznego formatu plików dla dowolnej aplikacji. Języki takie nazywamy aplikacjami XML. Takich języków jest coraz więcej, najbardziej znane to:

- XHTML – tworzenie stron internetowych;
- MathML – tworzenie dokumentów zawierających formuły matematyczne;
- WML – tworzenie stron WAP dla urządzeń mobilnych;
- GedML – obsługa danych genealogicznych;
- MusicML – notacja muzyczna;
- VoiceML – dane głosowe;
- ThML – teksty teologiczne;
- SVG – grafika wektorowa;
- BITS – język technologii bankowych;
- XMLNews – wymiana aktualnych wiadomości;
- PDML – opis produktów;
- TEI – kodowanie tekstu wraz z informacją o jego treści.

Jak widać takich języków jest sporo i są one wykorzystywane praktycznie we wszystkich dziedzinach życia i nauki.

Zaletami XML'a jest to, że jest:[3]

- otwarty – nie jest obwarowany żadnymi patentami;
- elastyczny – oddzielony od treści, dlatego można go wykorzystać do dowolnych celów;
- bezpłatny;
- niezależny od platformy sprzętowej;
- można określać dane wraz z ich strukturą;
- dokumenty XML są czytelne dla ludzi;
- możliwość dokładnej kontroli poprawności składni dokumentów XML;
- duża ilość narzędzi do przetwarzania dokumentów XML na inne formaty.

Jednak jak wszystko XML posiada też swoje wady:[4]

- składnia jest nadmiarowa (szczególnie dla danych tabelarycznych) w porównaniu do reprezentacji binarnej;

---

3 *http://serwisy.blogspot.com/2008/07/serwisy-webowe-xml.html*

4 *http://serwisy.blogspot.com/2008/07/serwisy-webowe-xml.html*

- nadmiar ten powoduje zwiększenie objętości dokumentu i zwiększenie zużycia zasobów przetważających;
- model hierarchiczny źle pasuje dla danych mających postać grafu i tabeli;
- przy parsowaniu dokumentów XML za pomocą reprezentacji DOM (Dokument Object Model) konieczne jest wczytanie całego dokumentu do pamięci. Technika ta jest popularna w językach obiektowych (C++ czy Java).

1.3.2. Pochodzenie XML

Język XML wywodzi się od SGML (Standard Generalized Markup Language – standardowy uogólniony język znaczników), który służy do opisu struktury różnego typu danych. „SGML w odróżnieniu od języków znaczników dedykowanych do konkretnych zastosowań (takich jak np. HTML), nie jest zbiorem określonych znaczników i reguł ich użytkowania, lecz ogólnym, nadrzędnym językiem służącym do definiowania dowolnych znaczników i ustalania zasad ich poprawnego użytkowania.”[5]

Języka SGML używa się praktycznie do dwóch celów: [6]

- precyzyjnego definiowania zbiorów znaczników przeznaczonych do konkretnych zastosowań – przykładem jest język HTML
- ujednolicania zasad pisania i przekazywania dokumentów tekstowych w obrębie dużych firm lub instytucji.

SGML posiada duże możliwości, ale też duże rozmiary i jest trudny do stosowania. W związku z tym powstał uproszczony standard – XML. Popularne stało się stwierdzenie, że XML oferuje 80% możliwości SGML przy dziesięciokrotnie łatwiejszym ich wykorzystaniu.

---

[5] J. Kowal, *Import i eksport danych hydrometeorologicznych z wykorzystaniem formatu XML oraz interfejsu SAX*, Praca magisterska, Politechnika Krakowska 2008 r.

[6] *tamże*

1.3.3. Struktura dokumentu

Listing 1.1. Przykładowy dokument XML. [Źródło opracowanie własne].

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!--Przykład xml-->
<osoby>
	<osoba usunieto="nie">
		<imie>Andrzej</imie>
		<nazwisko>Jurzec</nazwisko>
		</telefon>
	</osoba>
	<osoba usunieto="tak">
		<imie>jaksnessd</imie>
		<nazwisko>kestaj</nazwisko>
		</telefon>
	</osoba>
</osoby>
```

1.3.3.1. Prolog

Listing 1.2. Przykładowy prolog XML. [Źródło: opracowanie własne].

```
<?xml version="1.0" encoding="iso-8859-2" standalone="no"?>
```

Prolog umieszcza się na początku dokumentu. Może zawierać różne deklaracja dotyczące przetwarzania dokumentu np.:

- version = "1.0" – jest wymaganym atrybutem i musi być umieszczony jako pierwszy, określa wersję XML, w jakiej stworzony jest dokument;
- "encoding" – określa kodowanie znaków;
- "standalone" – określa, czy dokument odwołuje się do źródeł zewnętrznych czy też nie.

1.3.3.2. Elementy

Każdy element musi posiadać znacznik otwierający <nazwa>, oraz zamykający </nazwa>, gdzie nazwa jest to konkretna nazwa danego elementu. Pomiędzy znacznikiem otwierającym, a zamykającym można umieszczać wartość danego elementu. Mogą to być konkretne dane, lub też inne elementy, dzięki czemu można budować strukturę drzewiastą dokumentu. Element może posiadać atrybuty, które definiowane są w znaczniku otwierającym np. <nazwa atr="wartosc"> Element pusty w skrócie można zapisać <nazwa/> Element nadrzędny dla wszystkich pozostałych elementów często jest nazywany korzeniem. W listingu 1.1. jest to element osoby.

1.3.3.3. Atrybuty

Atrybut dodawany jest do elementu i służy do szczegółowego opisu danego elementu.

Składnia atrybutu wygląda następująco:

name="value", gdzie:

name jest to konkretna nazwa atrybutu,

value jest wartością tego atrybutu. Wartość zawsze musi być ujęta w cudzysłów (pojedynczy lub podwójny).

1.3.3.4. Komentarze

Listing 1.3. Komentarz XML. [Źródło: opracowanie własne].

```
<!--Wszystko, co jest tutaj napisane nie zostanie wzięte pod uwagę -->
```

Komentarz rozpoczyna się ciągiem znaków "<!--", a kończy "-->". Wszystko, co jest pomiędzy nie jest brane pod uwagę podczas przetwarzania dokumentu. Może służyć np. do dodawania opisów, lub usuwania aktualnie niepotrzebnej części kodu.

1.4. SVG

SVG (Scalable Vector Graphics) jest aplikacją języka XML służącą do opisu dwuwymiarowej, statycznej i animowanej grafiki wektorowej. Został stworzony przez W3C (World Wide Web Consortium), które zajmuje się tworzeniem standardów dla sieci Web.

SVG jest otwartym standardem, dzięki czemu dostęp do niego nie jest kontrolowany przez żadną firmę i może być swobodnie wykorzystywany w aplikacjach.

1.4.1. Historia SVG

Początki historii SVG sięgają roku 1998, kiedy to w listopadzie po raz pierwszy publicznie udostępniono wymagania, jakie będą stawiane dla SVG. 11 lutego 1999r. ukazała się pierwsza wersja specyfikacji SVG 1.0., która rekomendację W3C uzyskała dopiero we wrześniu 2001r.

Po ukończeniu SVG 1.0. przyszedł czas na dalszy rozwój i specyfikację SVG 1.1., której największą zmianą w stosunku do poprzedniej wersji było wprowadzenie modularyzacji, czyli podzieleniu całości specyfikacji na niezależne od siebie moduły. Umożliwiło to, poza tworzeniem pełnej wersji specyfikacji, tworzenie specyfikacji uproszczonych przeznaczonych tylko do konkretnych celów i posiadających pewne ograniczenia. Tak powstały specyfikacje SVG 1.1. Tiny (przeznaczone dla telefonów komórkowych), oraz SVG 1.1. Basic (przeznaczone dla palmtopów), w skrócie obie te specyfikacje nazywane są SVG Mobile (październik 2001r.)

W kwietniu 2002 ukazały się wymagania, jakie będą stawiane dla specyfikacji SVG 1.1./1.2./2.0.

W styczniu 2003r. specyfikacja SVG 1.1. oraz SVG 1.1. Mobile uzyskały rekomendację W3C.

Kolejny krok to prace nad specyfikacją 1.2., od której to wersji powstały dwie dodatkowe specyfikacje SVG 1.2. Print dotycząca wydruków, oraz SVG 1.2. Filters dotycząca efektów graficznych .

Jak do tej pory tylko specyfikacja SVG 1.2. Tiny uzyskała rekomendację W3C (grudzień 2008).

1.4.2. SVG skompresowany – SVGZ

Dokumenty SVG mogą posiadać rozszerzenia svg, lub svgz (SVG skompresowany w formacie gzip). Jedyną zaletą plików svgz jest zmniejszenie rozmiaru w stosunku do svg o ok. 50 %.

Główne wady svgz to:

- pliki svgz nie są czytelne dla człowieka;
- pliki svgz nie mogą być przeszukiwane zwykłymi algorytmami do przeszukiwania tekstu;
- dodatkowy nakład pracy na kompresję(zapis) i dekompresję (odczyt) pliku;
- nie wszystkie programy obsługują pliki svgz (np. Mozilla Firefox).

2. Specyfikacja SVG

2.1. Wprowadzenie

W rozdziale tym zostanie opisany fragment specyfikacji SVG 1.1. Starałem się tutaj wybrać najistotniejsze elementy tej specyfikacji, które są najczęściej wykorzystywane przy tworzeniu grafiki wektorowej.

2.1.1. Typy danych użyte w opisie specyfikacji

- <integer> – liczba naturalna z zakresu -2147483648 – 2147483647;
- <number> – liczba rzeczywista z zakresu $-3.4 * 10^{38} – 3.4 * 10^{38}$;
- <length> – odległość w układzie współrzędnych w podanej jednostce; jeżeli żadna jednostka nie zostanie podana, użyta zostanie jednostka domyślna danego układu;
- <coordinate> – jest to odległość od początku układu współrzędnych wzdłuż określonej osi;
- <angle> – jest to miara kąta, może przyjmować wartości z zakresu <number>. Dozwolone jednostki to:
    - deg – stopnie,
    - grad – grady,
    - rad – radiany,

  jeżeli jednostka nie zostanie podana przyjmowane są stopnie;
- <color> – bezpośrednio podana wartość koloru jako nazwa koloru, lub jako wartość liczbowa:
    - **#f00** – każda ze składowych r, g, b, zapisana jest za pomocą jednej cyfry w systemie szesnastkowym;
    - **#ff0000** – każda ze składowych r, g, b, zapisana jest szesnastkowo za pomocą dwóch cyfr;

- **rgb(255,0,0)** – składowe r, g, b, opisane są liczbami naturalnymi z zakresu 0 – 255;
- **rgb(100%, 0%, 0%)** – składowe r, g, b, opisane są wartością procentową z zakresu 0.0% – 100.0%;

- <list of xxx> (gdzie xxx reprezentuje dowolny typ) – jest to lista wartości danego typu oddzielonych od siebie przecinkami i/lub białymi znakami (np. spacja, tabulator, znak nowej linii).

2.1.2. Jednostki

Wykaz jednostek, jakie mogą być użyte w dokumentach SVG:

- pt – punkt – 1pt = 1/72 cala;
- % – procent wielkości okna, w którym wyświetlany jest obiekt (viewbox);
- pc – pica – 1pc = 1/6 cala;
- em – firet – stopień pisma 1 em = wysokość stopnia pisma „em to jednostka względna, zależna od wysokości stopnia pisma (rozmiaru czcionki). 1 em ma długość równą stopniowi pisma (rozmiarowi czcionki). Jest to jednostka idealnie nadająca się do określania parametrów tekstu takich jak: stopień pisma, wielkość interlinii, odstępów międzywyrazowych i międzyliterowych, a w szczególności atrybutów odnoszących się do szerokości znaku”[7];
- px – piksel – „piksel to jednostka zależna od rozdzielczości urządzenia (ekranu, drukarki itp.). Rozmiar określony w pikselach nie zależy od rozmiaru innych elementów” [8];
- ex – wysokość x (x-height) – „1 ex = x (wysokość małej litery bez wydłużeń) – ex to jednostka względna, zależna od wysokości małych liter w danym kroju. 1 ex ma długość równą wysokości małych liter (bez wydłużeń dolnych) w danym kroju (np. litera „x” — stąd nazwa x-height). Jak widać, wielkość ex jest zależna od rodzaju użytego kroju. Każdy krój ma inny stosunek wysokości dużych liter do małych. Średnio ten stosunek wynosi 1:2” [9];
- mm – milimetry;

---

7 *http://taat.pl/typografia/jednostki.html*

8 *http://taat.pl/typografia/jednostki.html*

9 *http://taat.pl/typografia/jednostki.html*

- in – cale;
- cm – centymetry.

Jeżeli nie zostanie podana żadna jednostka domyślnie przyjmowane są piksele.

2.2. Struktura dokumentu

2.2.1. Nagłówek dokumentu

Na początku dokumentu powinniśmy umieścić deklarację XML, która specyfikuje wersję języka XML użytą w dokumencie. Ponadto powinniśmy zamieścić odpowiednią wersję prologu:

Listing 2.1. Nagłówek dokumentu SVG. [Źródło: opracowanie własne].

```
<?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">
```

Pierwsza linia zawiera deklarację XML. Atrybut standalone określa, czy dokument jest samodzielnym dokumentem, czy też zawiera odwołanie do pliku zewnętrznego. **standalone** = "no" oznacza, że ten dokument ma odwołanie do pliku zewnętrznego - w tym przypadku do odpowiedniego dla SVG DTD (DOCTYPE declaration).

Druga i trzecia linia to odwołanie do zewnętrznego DTD dla SVG. Jest on umieszczony pod adresem "**http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**". DTD znajduje się na stronie W3C i zawiera spis wszystkich dostępnych elementów SVG.

2.2.2. Element svg

Dokument SVG powinien składać się przynajmniej z jednego elementu 'svg'. Taki dokument może być niezależnym i samodzielnym dokumentem, lub może być zawarty w linii jako fragment głównego dokumentu XML.

Ważniejsze atrybuty:

- xmlns[:prefix] = "resource-name" – standardowy atrybut XML, określający przestrzeń nazw, dla SVG jest to: "http://www.w3.org/2000/svg";
- version = "<number>" – wskazuje wersję języka SVG, w której jest stworzony dokument;
- width = "<length>" – szerokość prostokątnego obszaru, w którym wyświetlany jest obiekt svg. Wartość ujemna jest błędna, wartość zerowa wyłącza rysowanie elementu. Wartość domyślna = "100%";
- height = "<length>" – Wysokość prostokątnego obszaru, w którym wyświetlany jest obiekt svg. Wartość ujemna jest błędna, wartość zerowa wyłącza rysowanie elementu. Wartość domyślna = "100%".

2.2.3. Grupowanie obiektów – element g

Element **g** pozwala na grupowanie innych elementów. Mogą to być elementy graficzne np. kształty, gradienty, wzorce, lub też inne elementy **g.** Grupowanie, w przypadku, gdy używane są również elementy **desc** i **title,** umożliwia dodanie informacji na temat struktury dokumentu i jego semantyki. Grupa elementów, podobnie jak pojedynczy obiekt, może mieć nadaną nazwę za pomocą atrybutu **id**.

2.2.4. Tworzenie szablonów – element defs i symbol

Elementy **defs** i **symbol** pozwalają tworzyć grupy elementów. W odróżnieniu od elementu **g** zawartość obu tych elementów nie jest bezpośrednio rysowana, renderowane jest dopiero odwołanie do obiektów zdefiniowanych za pomocą **defs** i **symbol**.

Różnica pomiędzy **defs**, a **symbol** polega na tym, że element **symbol** posiada atrybuty **viewBox** i **preserveAspectRatio**, które pozwalają na dopasowanie do prostokątnego widoku zdefiniowanego w odwołującym się elemencie **use**.

Użycie tych elementów dla grafik, które są wykorzystywane wiele razy w dokumencie, poprawia ich czytelność.

2.2.5. Wstawianie metadanych – elementy desc i title

Każdy element zbiorczy i graficzny w formacie SVG może posiadać element **desc** lub/i **title**. Umożliwiają dodanie do dokumentu dowolnych informacji tekstowych, nie powodując zmian w wyświetlaniu dokumentu. Elementy te nie są renderowane, ale programy mogą wyświetlać tekst zawarty w elemencie **title,** jako podpowiedź, kiedy urządzenie wskazujące porusza się po danym elemencie.

2.2.6. Element use

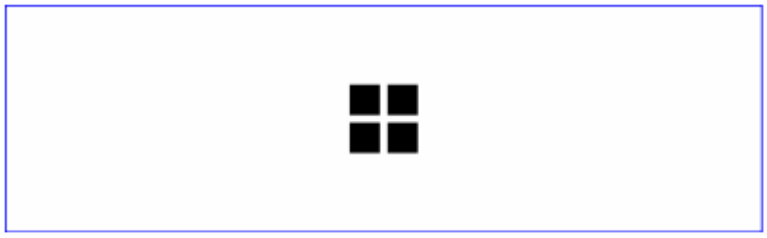
Każdy element **'svg'**, **'symbol'**, **'g'**, graficzny element, lub inny element **'use'** jest potencjalnie szablonem, który może być powtórnie wykorzystany za pomocą elementu **'use'**. Element **use** odnosi się do innego elementu i powoduje, że zawartość graficzna tego elementu jest załączona/narysowana w danym miejscu dokumentu. Element ten ma opcjonalne atrybuty **x**, **y**, **width** i **height**, które są używane do dopasowania zawartości graficznej, do prostokątnego obszaru aktualnego układu współrzędnych.

Dostępne atrybuty:

- x = "<coordinate>" – x–owa współrzędna pierwszego rogu prostokątnego obszaru, w który element będzie wstawiony, jeśli wartość nie jest podana, przyjęte będzie zero;
- y = "<coordinate>" – y–owa współrzędna pierwszego rogu prostokątnego obszaru, w który element będzie wstawiony, jeśli wartość nie jest podana, przyjęte będzie zero;
- width = "<length>" – szerokość prostokątnego obszaru, w który element będzie wstawiony, wartość ujemna generuje błąd, a wartość zerowa wyłącza renderowanie obiektu;
- height = "<length>" – szerokość prostokątnego obszaru, w który element będzie wstawiony, wartość ujemna generuje błąd, a wartość zerowa wyłącza renderowanie obiektu;
- xlink:href = "<uri>" – odnośnik URI do obiektu który ma zostać użyty.

Listing 2.2. Przykład użycia elementu use [Źródło: http://www.w3.org].

```
<?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="10cm" height="3cm" viewBox="0 0 100 30" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<desc>Example Use02 - 'use' on a 'symbol'</desc>
<defs>
<symbol id="MySymbol" viewBox="0 0 20 20"><desc>MySymbol - four rectangles in
a grid</desc>
<rect x="1" y="1" width="8" height="8"/>
<rect x="11" y="1" width="8" height="8"/>
<rect x="1" y="11" width="8" height="8"/>
<rect x="11" y="11" width="8" height="8"/>
</symbol>
</defs>
<rect x=".1" y=".1" width="99.8" height="29.8"
fill="none" stroke="blue" stroke-width=".2" />
<use x="45" y="10" width="10" height="10"
xlink:href="#MySymbol" />
</svg>
```



Rys. 2.1 Przykład użycia elementu use [Źródło: http://www.w3.org].

2.3. Układy współrzędnych, widoki i transformacje

2.3.1. Układy współrzędnych

Obszar, po którym można rysować w SVG – tzw. płótno (canvas) jest nieskończenie wielki, jednak obszar wyświetlania obrazu jest ograniczony do prostokątnego obszaru – tzw. okno widoku (viewport). Klient użytkownika

(user agent) – aplikacja, która wyświetla dokument SVG, musi ustalić, w jakim obszarze dokument SVG będzie wyświetlany.

W SVG rozróżniane są dwa układy współrzędnych: lokalny układ współrzędnych (user coordinate system), oraz układ współrzędnych widoku (viewport coordinate system). Punkt (0 0) znajduje się w lewym górnym rogu, współrzędna x rośnie w prawą stronę, natomiast współrzędna y rośnie do dołu. Na początku oba te układy pokrywają się, jednak definiowanie transformacji lub okien widoku (**viewBox**), powoduje powstawanie nowych układów, które mogą być względem siebie w dowolny sposób przekształcone, w zależności od przyjętych transformacji lub widoków.

2.3.2. Widoki

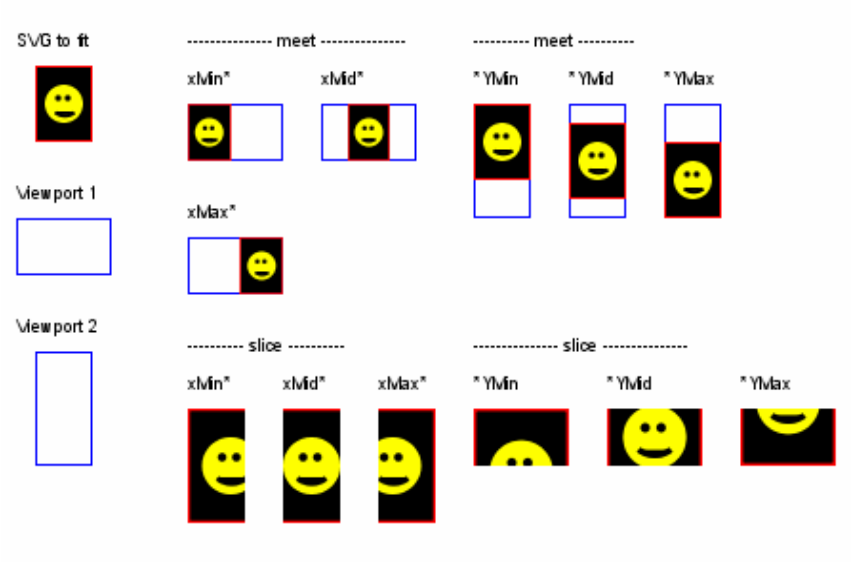
Czasami konieczne jest wyświetlenie grafiki w określonym obszarze (**viewPort**). Atrybut **viewBox** (kontener widoku) można stosować dla elementów **svg**, **symbol**, **use**, **image**, **foreignObject**, **marker**, **pattern**, oraz **view**. Atrybut ten pozwala zdefiniować wielkość kontenera, w którym będzie wyświetlany dany element, bądź grupa elementów. Wartość atrybutu viewBox jest to lista czterech liczb "x y szerokość wysokość" oznaczających położenie i wielkość obszaru, jaki będzie widoczny. Wartość ujemna wysokości lub szerokości jest błędna, natomiast wartość zero wyłączy rysowanie obiektu. Obszar ten zostanie automatycznie dopasowany do wielkości elementu, dla którego jest zdefiniowany atrybut viewBox.

Atrybut **preserveAspectRatio** pozwala nam kontrolować, w jaki sposób kontener widoku zostanie dopasowany do obszaru wyświetlania elementu.

preserveAspectRatio = "<align> <adjust>"

- align = "xMinYMin | xMidYMin | xMaxYMin | xMinYMid | xMidYMid | xMaxYMid | xMinYMax | xMidYMax | xMaxYMax" – Wartości Min, Mid i Max oznaczają odpowiednio wyrównanie zawartości viewBox do początku, środka i końca obszaru, względem odpowiedniej osi;
- adjust = "meet | slice" – określa sposób dopasowania okna widoku do wielkości obiektu:
    - **meet** powoduje dopasowanie dłuższego boku viewBox do krótszego boku obszaru elementu. Efekt jest taki, że cały obszar viewBox jest widoczny, mogą pozostać niewypełnione miejsce w obszarze elementu;

- **slice** powoduje dopasowanie krótszego boku viewBox do dłuższego boku obszaru elementu. Cały obszar elementu zostanie wypełniony, jednak część widoku viewBox nie zostanie pokazana.



Rys. 2.2. Przykłady dopasowania widoku (viewBox) do obszaru (viewPort) [Źródło: http://www.w3.org].

2.3.3. Transformacje

Transformacje są to dowolnego typu przekształcenia geometryczne, które mogą zostać opisane za pomocą macierzy przekształcenia (zob. 2.3.3.1.) . Definiowane są one za pomocą atrybutu **transform,** który jako wartość, przyjmuje dowolnie długą kombinację komend opisanych poniżej. Kolejne transformacje wykonywane są w kolejności ich podania.

2.3.3.1. Macierz przekształcenia

Matrix(a b c d e f) – macierz przekształcenia, która definiuje transformację w następujący sposób:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Za pomocą macierzy przekształcenia, można opisać wszystkie inne transformacje.

2.3.3.2. Przesunięcie

Translate(tx [ty]) – przesunięcie o wartość tx w osi x i o ty w osi y. Jeżeli ty nie zostanie podane przyjmowana jest wartość 0. translate(tx ty) ≡ matrix(1 0 0 1 tx ty)

Listing 2.3. Przesunięcie obiektu w osi x i osi y. [Źródło: opracowanie własne].

```
<?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="4cm" height="4cm" viewBox="-100 -100 400 400" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<rect id="r" x="10" y="10" width="200" height="200"/>
</defs>
<use xlink:href="#r" fill="lightgrey"/>
<use xlink:href="#r" transform="translate(50,50)" fill="red"/>
</svg>
```

Rys. 2.3. Przesunięcie obiektu w osi x i osi y. [Źródło: opracowanie własne].

2.3.3.3. Skalowanie

Scale(sx [sy]) – przeskalowanie rozmiaru w osi x o wartość sx i w osi y o sy. Jeżeli sy nie zostanie podane przyjmowane jest takie, jak wartość sx. Dla wartości mniejszych od zera wystąpi dodatkowo odbicie lustrzane względem odpowiadającej osi.

Scale(sx sy) ≡ matrix(sx 0 0 sy 0 0).

Listing 2.4. Skalowanie obiektu w osi x i osi y. [Źródło: opracowanie własne].

```
<use xlink:href="#r" fill="lightgrey"/>
<use xlink:href="#r" transform="scale(2,0.5)" fill="red"/>
```

Rys. 2.4. Skalowanie obiektu w osi x i osi y. [Źródło: opracowanie własne].

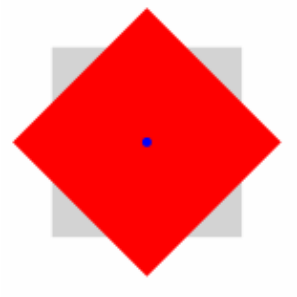
2.3.3.4. Obrót

rotate(kąt [cx cy]) – obrót o podany kąt wokół punktu (cx cy). Jeżeli punkt obrotu nie zostanie podany, obrót zostanie wykonany względem początku lokalnego układu współrzędnych.

Rotate(kąt sx sy) ≡ matrix(cos(a) sin(a) -sin(a) cos(a) 0 0) translate(cx cy).

Listing 2.5. Obrót obiektu względem podanego punktu. [Źródło: opracowanie własne].

```
<use xlink:href="#r" fill="lightgrey"/>
<use xlink:href="#r" transform="rotate(45 100 100)" fill="red"/>
<circle cx="100" cy="100" r="5" fill="blue" />
```



Rys. 2.5. Obrót obiektu względem podanego punktu. [Źródło: opracowanie własne].

2.3.3.5. Przekrzywienia

- skewX (kąt) – przekrzywienie w osi x. Jest to zmiana współrzędnych x według wzoru: x' = x + y * tan(kąt).
  skewX(kąt) ≡ matrix(1 0 tan(a) 1 0 0);

  Listing 2.6. Przekrzywienie obiektu względem osi x. [Źródło: opracowanie własne].

```
<use xlink:href="#r" fill="lightgrey"/>
<use xlink:href="#r" transform="skewX(25)" fill="red"/>
```

Rys. 2.6. Przekrzywienie obiektu względem osi x. [Źródło: opracowanie własne].

- skewY (kąt) – przekrzywienie w osi y. Jest to zmiana współrzędnych y według wzoru: y' = y + x * tan(kąt).
  skewY(kąt) ≡ matrix (1 tan(a) 0 1 0 0);

Listing 2.7. Przekrzywienie obiektu względem osi y. [Źródło: opracowanie własne].

```
<use xlink:href="#r" fill="lightgrey"/>
<use xlink:href="#r" transform="skewY(25)" fill="red"/>
```

Rys. 2.7. Przekrzywienie obiektu względem osi y. [Źródło: opracowanie własne].

2.4. Ścieżki

Ścieżki (paths) reprezentują figurę, która może zostać wypełniona, obrysowana, lub użyta, jako ścieżka wycinająca. Rysowanie za pomocą path jest bardzo intuicyjne i opiera się na koncepcji aktualnego punktu (current point).

Za pomocą komendy moveto możemy ustawić aktualny punkt bez rysowania linii, skąd możemy poprowadzić prostą, bądź krzywą do następnego punktu, który automatycznie stanie się punktem aktualnym. Czynności te możemy wielokrotnie powtórzyć tworząc dowolnie skomplikowaną figurę. Na końcu można użyć komendy closepath (Z), która zamknie nam ścieżkę tworząc linię prostą do ostatniej pozycji moveto.

2.4.1. Atrybuty elementu path

pathLength = "<number> – całkowita długość ścieżki. Służy do kalibracji długości ścieżki z długością obliczoną przez program. Wykorzystywane jest to np. przy umieszczaniu tekstu.

d = "<path data>" – definicja kształtu ścieżki. Może przyjmować komendy spośród: moveto, line, curve, arc i closepath.

Każda komenda ma dwie wersje:

- napisana z dużej litery oznacza bezwzględne położenie punktu, czyli w odniesieniu do początku układu współrzędnych;
- napisana z małej litery oznacza względne położenie punktu, czyli w odniesieniu do aktualnego punktu.

W każdej z komend podajemy punkt, do którego linia ma być narysowana. Linia jest rysowana od punktu aktualnego do podanego. Po wykonaniu komendy punkt, do którego została narysowana linia staje się nowym punktem aktualnym.

2.4.1.1. Otwarcie ścieżki

moveto – M (x, y), lub m (x,y) – tworzy nową pod-ścieżkę o punkcie początkowym(x,y). Jeśli po komendzie następuje wiele par współrzędnych, to są one traktowane, jako niejawne komendy "lineto".

2.4.1.2. Zamknięcie ścieżki

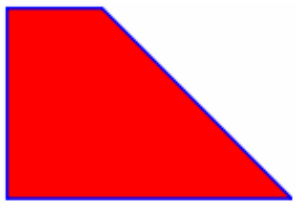
closepath – Z, lub z – zamyka bieżącą pod-ścieżkę rysując linię prostą do jej punktu początkowego. Jeżeli zaraz po komendzie closepath nastąpi komenda rysowania linii to punkt początkowy aktualnej pod-ścieżki stanie się punktem początkowym nowej pod-ścieżki.

2.4.1.3. Linie proste

- lineto – L(x, y), lub l(x, y) – rysuje linię prostą z aktualnego punktu do podanego. Podany punkt staje się nowym aktualnym punktem;
- horizontal lineto – H(x), lub h(x) – rysuje poziomą linię z aktualnego punktu (cpx, cpy) do punktu (x, cpy);
- vertical lineto. – V(x), lub v(x) – rysuje pionową linię z aktualnego punktu (cpx, cpy) do punktu (cpx, y).

Listing 2.8. Przykład rysowania linii prostych. [Źródło: opracowanie własne].

```
<?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="4cm" height="4cm" viewBox="0 0 400 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<path d="M 100 100 v 200 h 300 l -200 -200 z" fill="red" stroke="blue"
stroke-width="3" />
</svg>
```



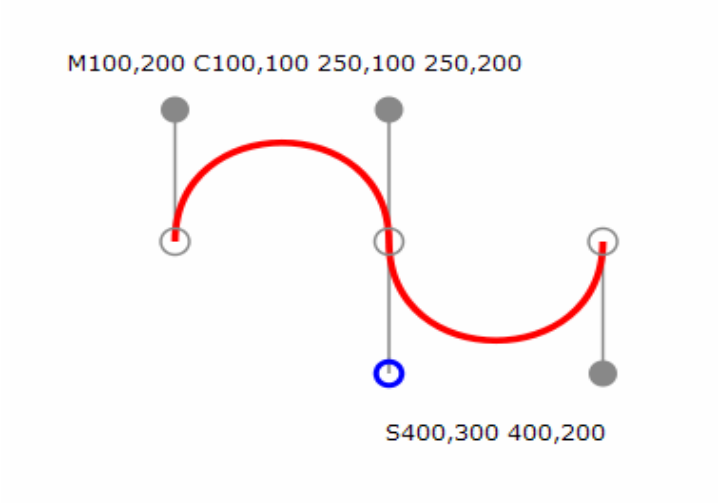
Rys. 2.8. Przykład rysowania linii prostych. [Źródło: opracowanie własne].

2.4.1.4. Krzywe Bézier'a trzeciego stopnia

- curveto – C(x1 y1 x2 y2 x y), lub c(x1 y1 x2 y2 x y) – rysuje krzywą Bézier'a trzeciego stopnia do punktu (x, y) używając punktu (x1, x2) jako punktu kontrolnego na początku krzywej oraz (x2, y2) jako punktu kontrolnego na końcu krzywej;
- shorthand/smooth curveto – S(x1 y1 x2 y2 x y), s(x1 y1 x2 y2 x y) – rysuje krzywą Bézier'a trzeciego stopnia z aktualnego punktu do (x,y). Punkt kontrolny jest przyjęty jako odbicie drugiego punktu kontrolnego w poprzedniej komendzie względem aktualnego punktu. Jeśli nie ma poprzedniej komendy lub nie było to C, c, S, s to punkt kontrolny pokrywa się z aktualnym punktem.

Listing 2.9. Przykład rysowania krzywych Bézier'a trzeciego stopnia. [Źródło: http://www.w3.org].

```
<?xmlversion="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="400" height="300" viewBox="0 0 500 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<g fill="none" stroke="#888888" stroke-width="2">
<polyline points="100,200 100,100" />
<polyline points="250,100 250,200" />
<polyline points="250,200 250,300" />
<polyline points="400,300 400,200" /></g>
<g fill="none" stroke="red" stroke-width="5">
<path class="SamplePath" d="M100,200 C100,100 250,100 250,200
S400,300 400,200" /></g>
<g fill="none" stroke="#888888" stroke-width="2">
<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" /></g>
<g fill="#888888" stroke="none">
<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" /></g>
<g fill="none" stroke="blue" stroke-width="4">
<circle class="AutoCtlPoint" cx="250" cy="300" r="9" /></g>
<g font-size="22" font-family="Verdana">
<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></g>
</svg>
```



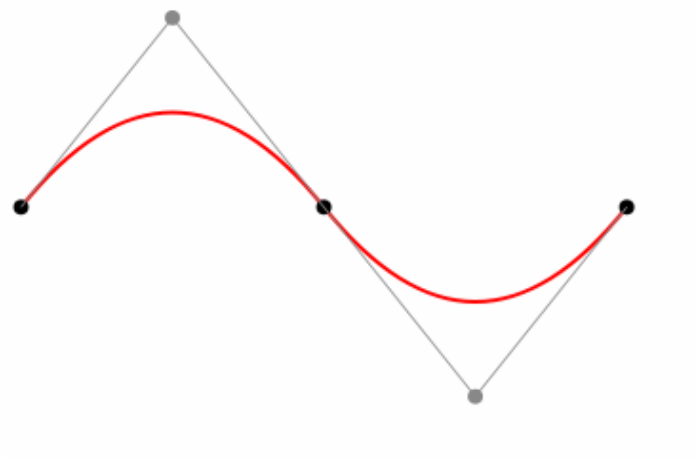
Rys. 2.9. Przykład rysowania krzywych Bézier'a trzeciego stopnia. [Źródło: http://www.w3.org].

2.4.1.5. Krzywe Bézier'a drugiego stopnia

- quadratic Bézier curveto – Q(x1 y1 x y), lub q(x1 y1 x y) – rysuje krzywą Bézier'a drugiego stopnia z aktualnego punktu do (x,y) używając (x1,y1) jako punktu kontrolnego;
- shorthand/smooth quadratic Bézier curveto – T(x y), lub t(x y) – rysuje krzywą Bézier'a drugiego stopnia z aktualnego punktu do (x, y). Punkt kontrolny jest przyjęty jako odbicie punktu kontrolnego w poprzedniej komendzie względem aktualnego punktu. Jeśli nie ma poprzedniej komendy lub nie było to C, c, S, s to punkt kontrolny pokrywa się z aktualnym punktem.

Listing 2.10. Przykład rysowania krzywych Bézier'a drugiego stopnia.
[Źródło: http://www.w3.org].

```
<?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="6cm" viewBox="0 0 1200 600"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<path d="M200,300 Q400,50 600,300 T1000,300"
fill="none" stroke="red" stroke-width="5" />
<g fill="black" >
<circle cx="200" cy="300" r="10"/>
<circle cx="600" cy="300" r="10"/>
<circle cx="1000" cy="300" r="10"/>
</g>
<g fill="#888888" >
<circle cx="400" cy="50" r="10"/>
<circle cx="800" cy="550" r="10"/>
</g>
<path d="M200,300 L400,50 L600,300
L800,550 L1000,300"
fill="none" stroke="#888888" stroke-width="2" />
</svg>
```



Rys. 2.10. Przykład rysowania krzywych Bézier'a drugiego stopnia. [Źródło: http://www.w3.org].

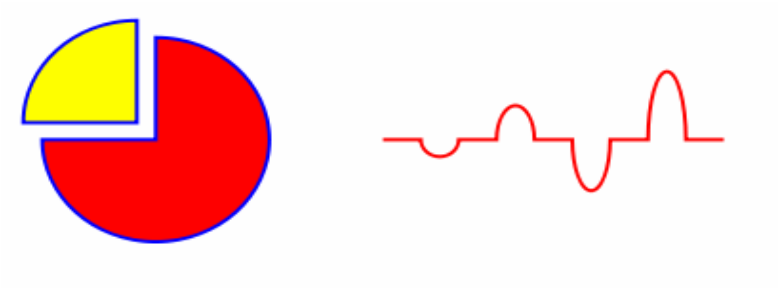
2.4.1.6. Wycinek koła

elliptical arc – a (rx, ry, x-axis-rotation, large-arc-flag, sweep-flag, x, y), lub **a** (rx, ry, x-axis-rotation, large-arc-flag, sweep-flag, x, y) – rysuje eliptyczny wycinek z aktualnego punktu do (x, y). Rozmiar i orientacja elipsy jest zdefiniowana poprzez dwa promienie (rx, ry) i x-axis-rotation, który wskazuje jak elipsa ma być obrócona. Środek elipsy (cx, cy) jest obliczany automatycznie, tak żeby pasował do innych podanych parametrów. large-arc-flag i sweep-flag pomagają określić jak wycinek będzie rysowany. Parametry large-arc-flag i sweep-flag mogą przyjmować wartość 0, lub 1, i określają, który wycinek koła będzie rysowany.

Rys. 2.11. Kombinacje parametrów large-arc-flag i sweep-flag. [Źródło: http://www.w3.org].

Listing 2.11. Przykład rysowania wycinków koła.[Źródło: opracowanie własne].

```
<?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="5.25cm" viewBox="0 0 1200 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<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 -150,150z"
fill="yellow" stroke="blue" stroke-width="5"/>
<path d="M600,200 l 50,0 a25,25 0 0,0 50,0 l 50,0
a25,50 0 0,1 50,0 l 50,0 a25,75 0 0,0 50,0 l 50,0
a25,100 0 0,1 50,0 l 50,0" fill="none" stroke="red" stroke-width="5"/>
</svg>
```



Rys. 2.12. Przykład rysowania wycinków koła.[Źródło: opracowanie własne].

2.5. Figury podstawowe

SVG zawiera następujący zestaw elementów opisujących figury podstawowe:

- prostokąty;
- okręgi;
- elipsy;
- linie;
- polilinie;
- wielokąty.

Wszystkie te kształty można również uzyskać wykorzystując element path. Kształty podstawowe mogą być wypełnione, obrysowane lub mogą posłużyć, jako kształt wycinający. Wszystkie właściwości dostępne dla elementu path, są również dostępne dla kształtów podstawowych.

2.5.1. Prostokąt

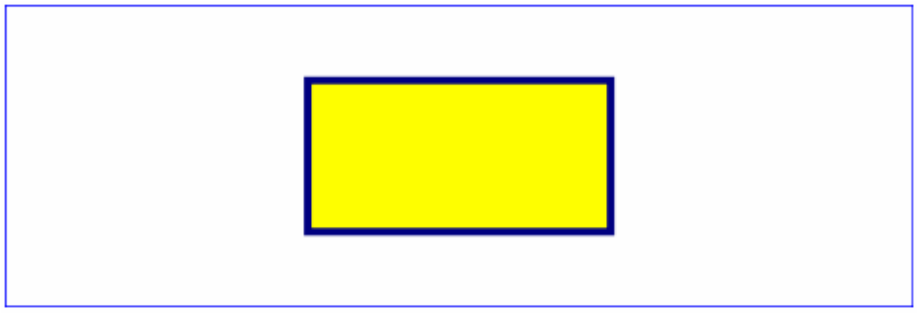
Element rect definiuje prostokąt, który jest zgodny z osiami lokalnego układu współrzędnych. Prostokąt może mieć zaokrąglone wierzchołki, poprzez ustawienie atrybutów rx i ry.

Dostępne atrybuty:

- **x** = "<coordinate>" – określa współrzędną x lewego dolnego rogu prostokąta. Wartość domyślna = "0";
- **y** = "<coordinate>" – określa współrzędną y lewego dolnego rogu prostokąta. Wartość domyślna = "0";
- **width** = "<length>" – określa szerokość prostokąta. Wartość mniejsza od zera jest błędna, wartość zero wyłącza rysowanie obiektu;
- **height** = "<length>" – określa wysokość prostokąta. Wartość mniejsza od zera jest błędna, wartość zero wyłącza rysowanie obiektu.
- **rx** = "<length>" – atrybut służy do zaokrąglania wierzchołków. Określa długość promienia w osi x elipsy użytej do zaokrąglenia. Wartość ujemna jest błędna;
- **ry** = "<length>" – atrybut służy do zaokrąglania wierzchołków. Określa długość promienia w osi y elipsy użytej do zaokrąglenia. Wartość ujemna jest błędna. Jeżeli została podana tylko jedna wartość rx lub ry, wartość drugiego atrybutu będzie taka sama. Jeżeli podana wartość jest większa niż połowa szerokości prostokąta, to przyjmowana jest wartość równa połowie szerokości prostokąta. Analogicznie jest dla rx.

Listing 2.12. Przykład użycia elementu rect. [Źródło: http://www.w3.org].

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2"/>
<rect x="400" y="100" width="400" height="200"
fill="yellow" stroke="navy" stroke-width="10" />
</svg>
```



Rys. 2.13. Przykład użycia elementu rect. [Źródło: http://www.w3.org].

2.5.2. Okrąg

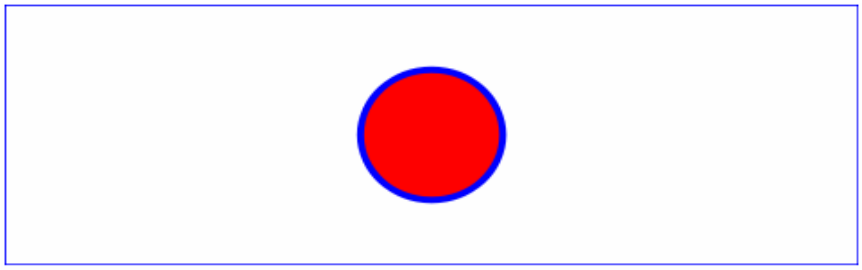
Element circle definiuje okrąg o podanym środku i promieniu.

Dostępne atrybuty:

- **cx** = "<coordinate>" – współrzędna x środka koła. Wartość domyślna = "0";
- **cy** = "<coordinate>" – współrzędna y środka koła. Wartość domyślna = "0";
- **r** = "<length>" – długość promienia okręgu. Wartość mniejsza od zera jest błędna, wartość równa zeru wyłącza rysowanie obiektu.

Listing 2.13. Przykład użycia elementu circle. [Źródło: http://www.w3.org].

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2"/>
<circle cx="600" cy="200" r="100" fill="red" stroke="blue" stroke-width="10" />
</svg>
```



Rys. 2.14. Przykład użycia elementu circle. [Źródło: http://www.w3.org].

2.5.3. Elipsa

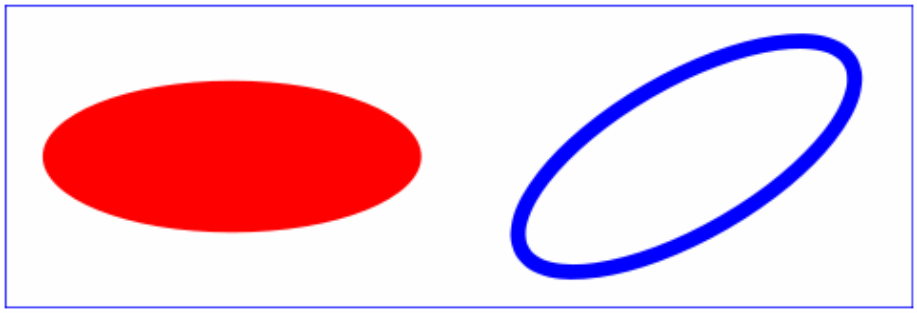
Element ellipse, definiuje elipsę której oś x jest zgodna o osią x lokalnego układu współrzędnych.

Dostępne atrybuty:

- **cx =** "<coordinate>" – współrzędna x środka elipsy. Jeżeli nie jest podana przyjmowana jest wartość 0;
- **cy** = "<coordinate>" – współrzędna x środka elipsy. Wartość domyślna = "0";
- **rx** = "<length>" – długość promienia w osi x. Wartość mniejsza od zera jest błędna, wartość równa zeru wyłącza rysowanie obiektu;
- **ry** = "<length>" – Długość promienia w osi y. Wartość mniejsza od zera jest błędna, wartość równa zeru wyłącza rysowanie obiektu.

Listing 2.14. Przykład użycia elementu ellipse. [Źródło: http://www.w3.org*].*

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2" />
<g transform="translate(300 200)">
<ellipse rx="250" ry="100" fill="red" />
</g>
<ellipse transform="translate(900 200) rotate(-30)" rx="250" ry="100"
fill="none" stroke="blue" stroke-width="20" />
</svg>
```



Rys. 2.15. Przykład użycia elementu ellipse. [Źródło: http://www.w3.org].

2.5.4. Linia

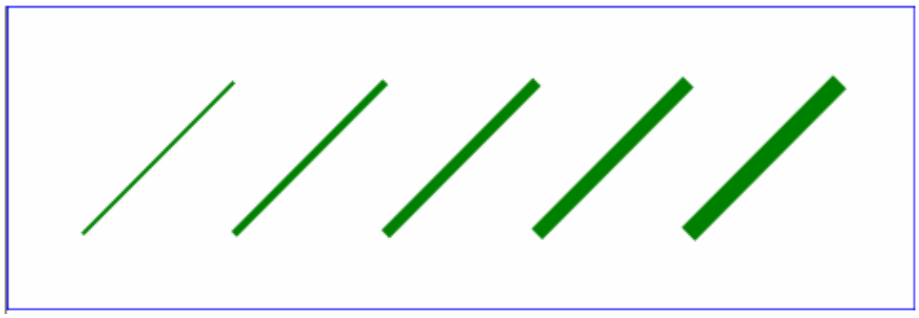
Element line definiuje linie prostą pomiędzy dwoma punktami.

Dostępne atrybuty:

- **x1** = "<coordinate>" – współrzędna x punktu początkowego. Wartość domyślna = "0";
- **y1** = "<coordinate>" – współrzędna x punktu początkowego. Wartość domyślna = "0";
- **x2** = "<coordinate>" – współrzędna x punktu końcowego. Wartość domyślna = "0";
- **y2** = "<coordinate>" – współrzędna y punktu końcowego. Wartość domyślna = "0".

Listing 2.15. Przykład użycia elementu line. [Źródło: http://www.w3.org].

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2" />
<g stroke="green" >
<line x1="100" y1="300" x2="300" y2="100" stroke-width="5" />
<line x1="300" y1="300" x2="500" y2="100" stroke-width="10" />
<line x1="500" y1="300" x2="700" y2="100" stroke-width="15" />
<line x1="700" y1="300" x2="900" y2="100" stroke-width="20" />
<line x1="900" y1="300" x2="1100" y2="100" stroke-width="25" />
</g>
</svg>
```



Rys. 2.16. Przykład użycia elementu line. [Źródło: http://www.w3.org].

2.5.5. Polilinia

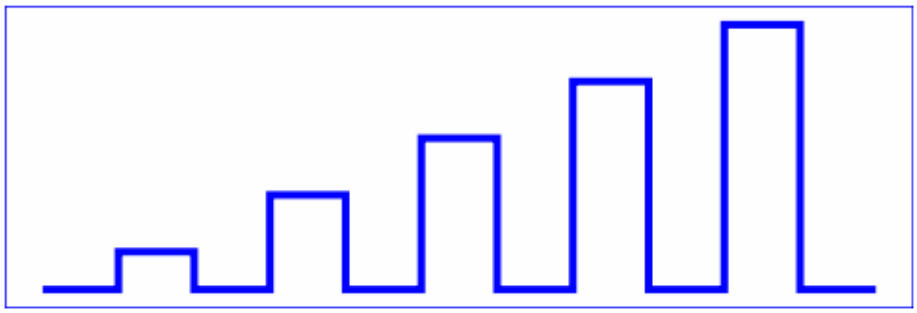
Element polyline definiuje zbiór połączonych linii prostych. Przeważnie są to figury otwarte.

Dostępne atrybuty:

points = "<list-of-points>" - punkty przez które przechodzi polilinia. Wszystkie współrzędne zdefiniowane są w lokalnym układzie współrzędnych. Nieparzysta ilość współrzędnych jest błędem.

Listing 2.16. Przykład użycia elementu polilinie. [Źródło: http://www.w3.org].

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2" />
<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" />
</svg>
```



Rys. 2.17. Przykład użycia elementu polilinie. [Źródło: http://www.w3.org].

2.5.6. Wielokąt

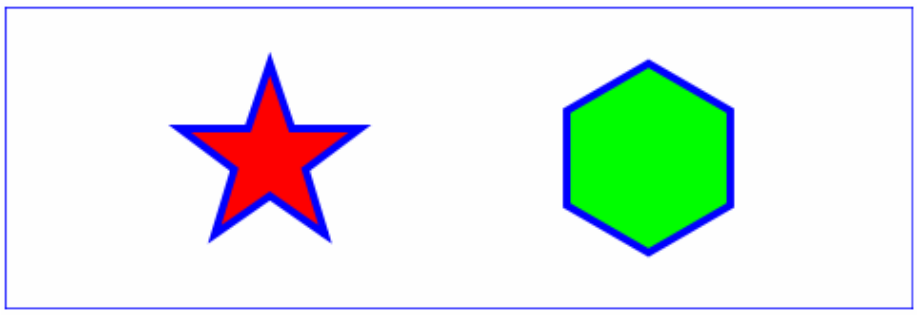
Element polygon definiuje zamkniętą figurę złożoną ze zbioru połączonych linii prostych.

Dostępne atrybuty:

points = "<list-of-points>" – punkty przez które przechodzi polilinia. Wszystkie współrzędne zdefiniowane są w lokalnym układzie współrzędnych. Nieparzysta ilość współrzędnych jest błędem.

Listing 2.17. Przykład użycia elementu polgon. [Źródło: http://www.w3.org].

```
<?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">
<rect x="1" y="1" width="1198" height="398"
fill="none" stroke="blue" stroke-width="2" />
<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" />
</svg>
```



Rys. 2.18. Przykład użycia elementu polygon. [Źródło: http://www.w3.org].

2.6. Tekst

Tekst do dokumentu SVG można dodać za pomocą elementu text. Jest on renderowany w taki sam sposób jak inne elementy graficzne, dlatego też przekształcenia (transformations), wypełnienia (painting), czy przycięcia (clipping) stosuje się do elementu text w taki sam sposób, jak dla podstawowych kształtów i ścieżek.

2.6.1. Element text

Element 'text' definiuje graficzny element zawierający tekst. Atrybuty i właściwości elementu 'text' wskazują na kierunek tekstu, czcionkę i atrybuty rysowania.

Do tekstu można stosować gradienty, wzorce, ścieżki wycinające, maski i filtry.

Dostępne atrybuty:

- x = "<list of coordinate>" – jeśli podana jest lista n współrzędnych, wtedy wartości te reprezentują położenie bezwzględne na osi x dla pierwszych n liter. Jeśli atrybut nie jest podany, to jego wartość przyjęta jest jako 0;
- y = "<list of coordinate>" – jeśli podana jest lista n współrzędnych, wtedy wartości te reprezentują położenie bezwzględne na osi y dla pierwszych n liter. Jeśli atrybut nie jest podany, to jego wartość przyjęta jest jako 0;
- dx = "list of length" – jeśli podana jest lista n długości, wtedy wartości te reprezentują przesunięcie wzdłuż osi x pierwszych n liter. Jeśli atrybut nie jest podany, to jego wartość przyjęta jest jako 0;
- dy = "list of length" – Jeśli podana jest lista n długości, wtedy wartości te reprezentują przesunięcie wzdłuż osi y pierwszych n liter. Jeśli atrybut nie jest podany, to jego wartość przyjęta jest jako 0.
- rotate = "<list of number>" – obrót poszczególnych liter tekstu o kąt w stopniach. Jeżeli podana jest tylko jedna wartość, każda litera z osobna zostanie obrócona. Jeżeli zostanie podanych n wartości, to tylko n kolejnych liter zostanie obróconych;
- textLength = "<length>" – długość tekstu obliczona przez autora. Całkowita długość ścieżki ustawiona przez autora. Służy do kalibracji długości ścieżki z długością obliczoną przez program. Jeśli atrybut jest podany, to wszystkie zmiany

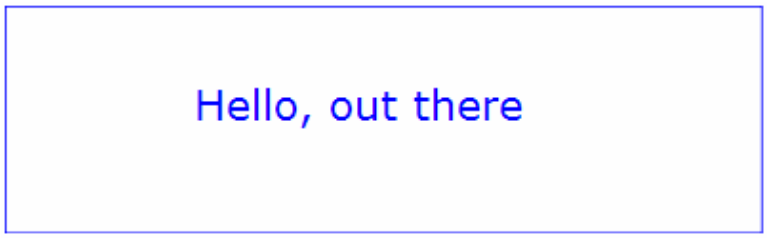
na tekście będą nadrzędnie kontrolowane przez tę wartość. Jeśli atrybut nie jest podany, przyjmowana jest wartość obliczona przez program;

- lengthAdjust = "spacing|spacingAndGlyphs" – podaje typ regulacji długości, którego użyje przeglądarka, aby renderowana długość pasowała do wartości podanej w atrybucie textLength;
    - spacing – zmieniane są tylko odstępy pomiędzy literami;
    - spacingAndGlyphs – zmieniane są odstępy oraz wielkość liter;

  Jeżeli atrybut nie został podany przyjmowana jest wartość sparing.

Listing 2.18. Przykład użycia elementu text. [Źródło: http://www.w3.org].

```
<?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="10cm" height="3cm" viewBox="0 0 1000 300"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<text x="250" y="150"
font-family="Verdana" font-size="55" fill="blue" >
Hello, out there
</text>
<rect x="1" y="1" width="998" height="298"
fill="none" stroke="blue" stroke-width="2" />
</svg>
```



Rys, 2.19. Przykład użycia elementu text. [Źródło: http://www.w3.org].

2.6.2. Element tspan

Wewnątrz elementu 'text' możemy zmienić właściwości tekstu, czcionki i aktualne położenie tekstu wstawiając element 'tspan'.

Atrybuty:

- x = "<list of coordinate>";
- y = "<list of coordinate>";
- dx = "list of length";
- dy = "list of length";
- rotate = "<list of number>";
- textLength= "list of length".

Definicja atrybutów jest taka sama jak dla elementu **text**. Atrybuty **x**, **y**, **dx**, **dy** i **rotate** elementu **'tspan'** są użyteczne w zaawansowanych grafikach z tekstem. W ten sposób możemy np. tworzyć teksty wielowierszowe.

Listing 2.19. Przykład użycia elementu tspan. [Źródło: http://www.w3.org].

```
<?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="10cm" height="3cm" viewBox="0 0 1000 300"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<g font-family="Verdana" font-size="45" >
<text fill="rgb(255,164,0)" >
<tspan x="300 350 400 450 500 550 600 650" y="100">
Cute and
</tspan>
<tspan x="375 425 475 525 575" y="200">
fuzzy
</tspan>
</text>
</g>
<rect x="1" y="1" width="998" height="298"
fill="none" stroke="blue" stroke-width="2" />
</svg>
```

Rys. 2.20. Przykład użycia elementu tspan. [Źródło: http://www.w3.org].

2.6.3. Element tref

Zawartość elementu 'text' może się znajdować bezpośrednio w tym elemencie, lub możemy użyć referencji do tekstu zdefiniowanego w innym miejscu za pomocą elementu tref.

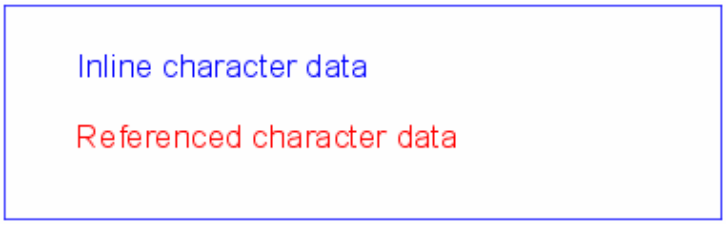
Dostępne atrybuty:

xlink:href = "<uri>" – adres URI (Uniform Resource Identifier) do fragmentu dokumentu SVG zawierającego informacje tekstowe, a którego zawartość będzie wstawiona do elementu 'tref'. Cała informacja tekstowa będzie wyrenderowana w danym miejscu.

Atrybuty x, y, dx, dy i rotate mają to samo znaczenie jak w przypadku elementu 'tspan'

Listing 2.20. Przykład użycia elementu tref. [Źródło: http://www.w3.org].

```
<?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="10cm" height="3cm" viewBox="0 0 1000 300"
xmlns="http://www.w3.org/2000/svg" version="1.1"
xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<text id="ReferencedText">
Referenced character data
</text>
</defs>
<text x="100" y="100" font-size="45" fill="blue" >
Inline character data
</text>
<text x="100" y="200" font-size="45" fill="red" >
<tref xlink:href="#ReferencedText"/>
</text>
<rect x="1" y="1" width="998" height="298"
fill="none" stroke="blue" stroke-width="2" />
</svg>
```



Rys. 2.21. Przykład użycia elementu tref. [Źródło: http://www.w3.org].

2.6.4. Element textPath

Za pomocą elementu **textPath,** można rozmieścić tekst wzdłuż elementu **path**.

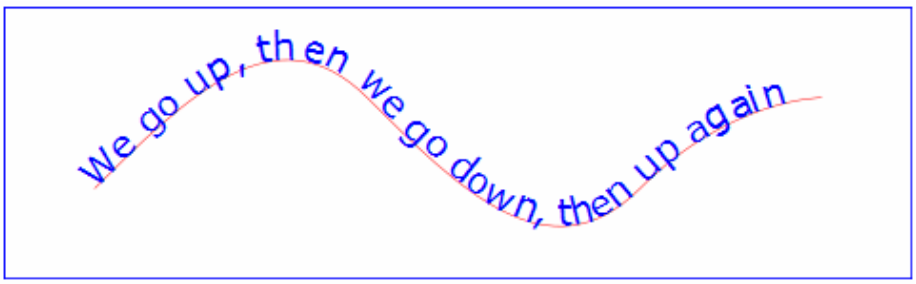
Dostępne atrybuty:

- startOffset = "<length>" – przesunięcie względem początku elementu path. Wartość domyślna = "0". Wartość mniejsza od zera jest błędem;
- method = "align | stretch" – określa sposób dopasowania tekstu do ścieżki:
  - **align** – dopasowanie następuje poprzez zmianę położenia i rotację poszczególnych liter tekstu;
  - **stretch** – dodatkowo może zmiana wielkości liter w celu lepszego dopasowania;

  wartość domyślna = "align".
- spacing = "auto | exact" – określa odstępy pomiędzy literami:
  - **auto** – odstępy są automatycznie regulowane w celu lepszego dopasowania;
  - **exact** – odstępy pomiędzy literami pozostają takie, jakie zostały zdefiniowane w tekście;
- xlink:href = "<uri>" – odwołanie uri do zdefiniowanego wcześniej elementu path, wzdłuż którego rozmieszczony zostanie tekst.

Listing 2.21. Przykład użycia elementu textPath. [Źródło: http://www.w3.org].

```
<?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="3.6cm" viewBox="0 0 1000 300" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<path id="MyPath"
d="M 100 200
C 200 100 300 0 400 100
C 500 200 600 300 700 200
C 800 100 900 100 900 100" />
</defs>
<use xlink:href="#MyPath" fill="none" stroke="red" />
<text font-family="Verdana" font-size="42.5" fill="blue" >
<textPath xlink:href="#MyPath" method="stretch">
We go up, then we go down, then up again
</textPath>
</text>
<rect x="1" y="1" width="998" height="298"
fill="none" stroke="blue" stroke-width="2" />
</svg>
```



Rys. 2.22. Przykład użycia elementu textPath. [Źródło: http://www.w3.org].

2.7. Rysowanie i wypełnianie

Elementy graficzne text, path oraz kształty podstawowe (np. okrąg, prostokąt) mogą zostać narysowane poprzez wypełnienie (fill), lub obrysowanie (stroke). Obiekt może zostać narysowany za pomocą:

- pojedynczego koloru;
- pojedynczego koloru z ustawioną przezroczystością;

- gradientu (stopniowe przejście pomiędzy dwoma kolorami) – liniowego lub kołowego;
- wzoru (pattern) – wektorowego lub rastrowego.

2.7.1. Wypełnianie

2.7.1.1. Właściwość fill

Określa sposób wypełnienia obiektu.

fill = "none | currentColor | <color> | <uri> | inherit"

- none – obiekt nie zostanie narysowany;
- currentColor – użyty zostanie kolor jaki został zdefiniowany w atrybucie **Color** obiektu;
- <color> – bezpośrednio podana wartość koloru;
- <uri> – odnośnik do zdefiniowanego wzorca, gradientu lub koloru;
- inherit – sposób wypełnienia dziedziczony jest z elementu nadrzędnego.

2.7.1.2. Właściwość fill-rule

Definiuje, w jaki sposób określane jest, czy punkt znajduje się wewnątrz obiektu, czy nie. Prowadzona jest półprosta z danego punktu w dowolnym kierunku i zliczana jest liczba przecięć z danym obiektem.

fill-rule = "nonzero | evenodd | inherit"

- nonzero – jeżeli liczba przecięć jest różna od zera to punk należy do figury;
- evenodd – jeżeli liczba przecięć jest nieparzysta to punk należy do figury;
- inherit – wartość jest dziedziczona z elementu nadrzędnego.

2.7.1.3. Właściwość fill-opacity

Określa przezroczystość wypełnienia obiektu.

fill-opacity = "<opacity-value> | inherit"

- <opacity-value> – wartość z zakresu "0.0" – "1.0". "1.0" oznacza obiekt całkowicie nieprzezroczysty, "0.0" – obiekt całkowicie przezroczysty;
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość domyślna = "1".

2.7.2. Obrysowywanie

2.7.2.1. Właściwość stroke

Określa sposób obrysowania obiektu.

stroke = "none | currentColor | <color> | <uri> | inherit".

- none – obiekt nie zostanie obrysowany;
- currentColor – użyty zostanie kolor, jaki został zdefiniowany w atrybucie Color obiektu;
- <color> – bezpośrednio podana wartość koloru;
- <uri> – odnośnik do zdefiniowanego wzorca, gradientu lub koloru;
- inherit –sposób obrysowania dziedziczony jest z elementu nadrzędnego.

2.7.2.2. Właściwość stroke-width

Grubość linii obrysowującej.

stroke-width "<length> | inherit"

- <length> – bezpośrednio podana wartość;
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość domyślna = "1".

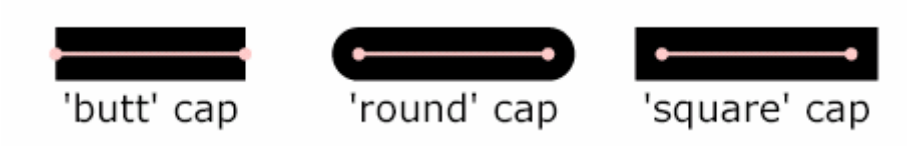
2.7.2.3. Właściwość stroke-linecap

Określa sposób zakańczania linii.

stroke-linecap ="butt | round | square | inherit".

- butt – zob. rys. 2.23;
- round – zob. rys. 2.23;

- square – zob. rys. 2.23;
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość domyślna = "butt".



Rys. 2.23. Rodzaje zakończeń linii. [Źródło: http://www.w3.org].

2.7.2.4. Właściwość stroke-linejoin

Określa sposób łączenia linii.

stroke-linejoin = "miter | round | bevel | inherit"

- miter – zob. rys. 2.24;
- round – zob. rys. 2.24;
- bevel – zob. rys. 2.24;
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość domyślna = "miter"

Rys. 2.24. Rodzaje łączeń linii. [Źródło: http://www.w3.org].

2.7.2.5. Właściwość stroke-miterlimit

Jeżeli linia ma ustawiony sposób łączenia na miter (ostre zakończenie), może się okazać, że długość złożenia (odległość pomiędzy wewnętrznym, a zewnętrznym wierzchołkiem) będzie nieproporcjonalnie dluga do grubości linii. Właściwość stroke-miterlimit określa maksymalny stosunek długości złożenia do grubości linii. Jeżeli

zostanie przekroczona ta wartość, zastosowane zostanie połączenie bevel (zakończenie ścięte).

stroke-miterlimit = "<miterlimit> | inherit"

- <miterlimit> – określa maksymalny stosunek długości złożenia wierzchołka do grubości linii. Musi to być liczba rzeczywista większa lub równa "1";
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość domyślna = "4".

2.7.2.6. Właściwość stroke-dasharray

Opisuje sposób rysowania linii przerywanych.

stroke-dasharray = "none | <list of numbers> | inherit"

- none – narysowana zostanie linia ciągła;
- <list of numbers> – jest to lista długości, która określa na przemian długość linii ciągłej i długość przerwy. Jeżeli zostanie podana nieparzysta liczba długości lista zostanie powtórzona tak żeby uzyskać liczbę parzystą. Wartości muszą być większe od "0";
- inherit – wartość dziedziczona z elementu nadrzędnego;
- wartość początkowa = "none".

2.7.2.7. Właściwość stroke-dashoffset

Określa przesunięcie wzoru linii przerywanych.

stroke-dashoffset = "<length> | inherit"

- <length> – wartość przesunięcia;
- wartość początkowa = "0".

2.7.2.8. Właściwość stroke-opacity

Określa przezroczystość linii obrysowującej.

stroke-opacity = "<opacity-value> | inherit"

- <opacity-value> – wartość z zakresu 0.0 – 1.0. 1.0 oznacza obiekt całkowicie nieprzezroczysty,0.0 – obiekt całkowicie przezroczysty;
- inherit – wartość dziedziczona z elementu nadrzędnego;
- Wartość domyślna = "1".

2.7.3. Znaczniki

2.7.3.1. Wprowadzenie

Znacznik jest to symbol graficzny, który może być przypisany do elementów:

- path;
- line;
- polyline;
- poligon.

Aby dodać znacznik do obiektu należy skorzystać z poniższych właściwości:

- **marker** = "none |inherit | <uri>" – dodaje znaczniki na początku, końcu i na łączeniu odcinków:
    - none – znacznik nie zostanie użyty;
    - inherit – wartość dziedziczona z elementu nadrzędnego;
    - <uri> – odnośnik do zdefiniowanego znacznika;
    - wartość domyślna = "none";
- marker-start = "none |inherit | <uri>" – dodaje znacznik na początku odcinka. Zob. właściwość marker;
- marker-end = "none |inherit | <uri>" – dodaje znacznik na końcu odcinka. Zob. właściwość marker;
- marker-mid = "none |inherit | <uri>" – dodaje znacznik na złączeniu odcinków. Zob. właściwość marker.

2.7.3.2. Element marker

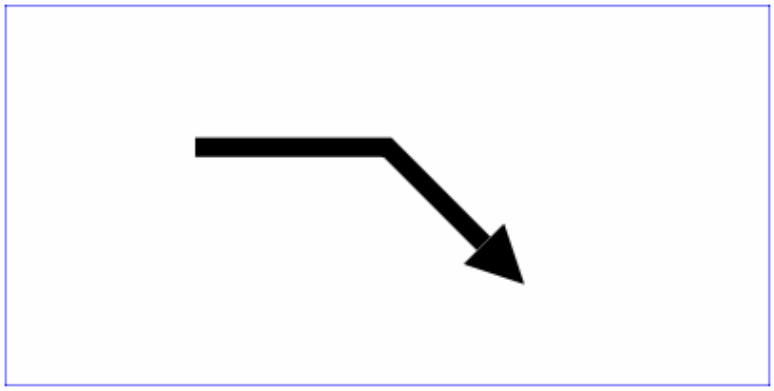
Element marker definiuje nowy znacznik. Nie jest rysowany bezpośrednio w miejscu w którym został zdefiniowany, lecz dopiero w miejscu w którym istnieje do niego odwołanie poprzez atrybuty marker-start, marker-end lub marker-mid.

Atrybuty:

- **markerUnits =** "strokeWidth | userSpaceOnUse" – definiuje układ współrzędnych dla atrybutów markerWidth i markerHeight:
    - strokeWidth – podana wartość będzie wielokrotnością grubości linii danego elementu;
    - userSpaceOnUse – podane wartości zostaną przyjęte w lokalnym układzie współrzędnych;
    - wartość domyślna = "strokeWidth";
- **refX** = "<coordinate>" – refX i refY określają współrzędne punktu, zgodnie z którym znacznik będzie. Wartość domyślna= "0";
- **refY** = "<coordinate>"– zob refX. wartość domyślna= "0";
- **markerWidth** = "<length>"– szerokość znacznika, zob. markerUnits;
- **markerHeight** = "<length>" – wysokość znacznika, zob markerUnits;
- **orient** = "auto | <angle>" – określenie, jak znacznik jest obrócony:
    - auto – obrót określany jest automatycznie, zgodnie z kierunkiem linii;
    - <angle> – znacznik obrócony jest o stałą wartość;
    - wartość domyślna = "0".

Listing 2.22. Przykład użycia elementu marker . [Źródło: http://www.w3.org].

```
<?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="4in" height="2in"
viewBox="0 0 4000 2000" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<defs>
<marker id="Triangle" viewBox="0 0 10 10" refX="0" refY="5"
markerUnits="strokeWidth" markerWidth="4"
markerHeight="3" orient="auto">
<path d="M 0 0 L 10 5 L 0 10 z" />
</marker>
</defs>
<rect x="10" y="10" width="3980" height="1980"
fill="none" stroke="blue" stroke-width="10" />
<path d="M 1000 750 L 2000 750 L 2500 1250"
fill="none" stroke="black" stroke-width="100"
marker-end="url(#Triangle)" />
</svg>
```



Rys. 2.25. Przykład użycia elementu marker. [Źródło: http://www.w3.org].

2.8. Gradienty i wzorce

W SVG możemy wypełniać lub obrysowywać obiekty używając:

- koloru;
- gradientu (liniowego lub kołowego);
- wzorce (pattern) (wektorowej lub rastrowej).

W tym celu stosujemy odwołanie URI we właściwościach '**fill**' lub '**stroke**'.

2.8.1. Gradienty

2.8.1.1. Wprowadzenie

Gradienty to stopniowe przechodzenie jednego koloru w drugi. Wyróżniamy dwa rodzaje:

- liniowe (linear gradients);
- kołowe (radial gradiets).

2.8.1.2. Gradienty liniowe

Gradienty liniowe definiuje się za pomocą elementu 'linearGradient'.

Dostępne atrybuty:

- gradientUnits ="userSpaceOnUse | objectBoundingBox" – Definiuje układ współrzędnych dla atrybutów x1, y1, x2, y2:
    - **userSpaceOnUse** – x1, y1, x2, y2 reprezentują współrzędne w lokalnym układzie współrzędnych, dla miejsca wstawiania gradientu;
    - **objectBoundingBox** – x1, y1, x2, y2 reprezentują współrzędne w układzie współrzędnych obiektu do którego gradient jest wstawiany;

  wartością domyślną jest "objectBoundingBox"w;
- gradientTransform = "<transform-list>" – lista przekształceń dla gradientu – zob. atrybut trans form;
- x1 ="<coordinate>" – x1, y1, x2, y2 definiują wektor gradientu (gradient vector), który określa długość oraz kierunek gradientu. Wartość może być podana jako liczba 0 – 1 lub jako procent 0 – 100%. Wartością domyślną jest 0%;
- y1 = "<coordinate>" – zob. x1. Wartością domyślną jest 0%;
- x2 = "<coordinate>" zob. x1. Wartością domyślną jest 100%;
- y2 = "<coordinate>" zob. x1. Wartością domyślną jest 0%;
- spreadMethod = "pad | reflect | repeat" – określa, co się stanie, gdy gradient nie wypełni całego obiektu, do którego zostanie wstawiony:
    - **pad** – zostaną użyte końcowe kolory do wypełnienia prostokąta;
    - **reflect** – gradient zostanie odbity;
    - **repeat** – gradient zostanie powtórzony;

  wartość domyślna to **pad**.

- xlink:href = "<uri>" – Odnośnik URI do innego gradientu liniowego lub kołowego zdefiniowanego w tym samym pliku SVG. Wszystkie argumenty nie zdefiniowane w aktualnym elemencie będą dziedziczone z tego gradientu.

Listing 2.23. Przykład użycia gradientu liniowego. [Źródło: http://www.w3.org].

```
<?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="8cm" height="4cm" viewBox="0 0 800 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<g>
<defs>
<linearGradient id="MyGradient">
<stop offset="5%" stop-color="#F60" />
<stop offset="95%" stop-color="#FF6" />
</linearGradient>
</defs>
<rect fill="none" stroke="blue" x="1" y="1" width="798" height="398"/>
<rect fill="url(#MyGradient)" stroke="black" stroke-width="5"
x="100" y="100" width="600" height="200"/>
</g>
</svg>
```

Rys. 2.26. Gradient liniowy. [Źródło: http://www.w3.org].

2.8.1.3 Gradient kołowy

Gradient kołowy zdefiniowany jest przez element radialGradient.

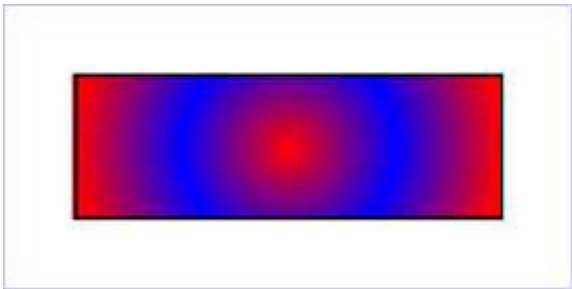
Dostępne atrybuty:

- gradientUnits ="userSpaceOnUse | objectBoundingBox" – zob gradient liniowy;
- gradientTransform = "<transform-list>" – lista przekształceń dla gradientu – zob. atrybut trans form;

- cx = "<coordinate>" – położenie środka okręgu w osi x. Wartość domyślna to "50%". Atrybuty cx, cy, i r definiują okrąg ograniczający obszar rysowania gradientu;
- cy = "<coordinate>" – położenie środka okręgu w osi y. Wartością domyślną jest "50%". Zob. cx;
- r = "<coordinate>" – promień okręgu. Wartością domyślna jest "50%". Zob cx;
- fx = "<coordinate>" – fx, fy definiują początkowy punkt gradientu. Wartością domyślną jest cx;
- fy = "<coordinate>" – zob. fx. Wartością domyślną jest cy;
- spreadMethod = "pad | reflect | repeat" – zob. gradient liniowy;
- xlink:href = "<uri>" – zob. gradient liniowy.

Listing 2.24. Przykład użycia gradientu kołowego[Źródło: http://www.w3.org].

```
<?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="8cm" height="4cm" viewBox="0 0 800 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<desc>Example radgrad01 - fill a rectangle by referencing a
radial gradient paint server</desc>
<g>
<defs>
<radialGradient id="MyGradient" gradientUnits="userSpaceOnUse"
cx="400" cy="200" r="300" fx="400" fy="200">
<stop offset="0%" stop-color="red" />
<stop offset="50%" stop-color="blue" />
<stop offset="100%" stop-color="red" />
</radialGradient>
</defs>
<!-- Outline the drawing area in blue -->
<rect fill="none" stroke="blue"
x="1" y="1" width="798" height="398"/>
<!-- The rectangle is filled using a radial gradient paint server -->
<rect fill="url(#MyGradient)" stroke="black" stroke-width="5"
x="100" y="100" width="600" height="200"/>
</g>
</svg>
```



Rys. 2.27. Gradient kołowy. [Źródło: http://www.w3.org].

2.8.1.4. Przejścia kolorów w gradientach

Kolory użyte w gradientach definiuje się za pomocą elementów stop, które mogą być wykorzystane wewnątrz elementów **linearGradient** i **radialGradient**.

Atrybuty:

offset = "<number> | <percentage>" – przesunięcie zmieniające się od 0 do 1 lub od 0% do 100%, które wskazuje gdzie ma nastąpić zmiana koloru w gradiencie. Dla gradientów liniowych, reprezentuje położenie na wektorze gradientu (gradient vector). Dla gradientów kołowych, reprezentuje odległość od punktu (fx,fy) do okręgu ograniczającego.

Element **stop** posiada dwie właściwości:

- **stop-color** = "<color>" – określa kolor w miejscu gradient stop;
- **stop-opacity** = "<opacity-value>" – określa przezroczystość.

2.8.2. Wzorce

Wzorzec jest to niewielki obiekt graficzny który może zostać użyty do wypełnienia określonego obszaru. Jeżeli obszar jest większy niż wzorzec, to wzorzec będzie powielony w pionie i/lub poziomie aż do momentu zapełnienia całego obszaru. Wzorzec powielany jest wzdłuż osi x i osi y aż do momentu wypełnienia

Wzorce definiowane są za pomocą elementu **pattern** Tak samo jak gradienty, wzorce dołączamy poprzez odwołanie URI we właściwościach **'fill'** lub **'stroke'**.

Dostępne atrybuty:

- patternUnits = "userSpaceOnUse | objectBoundingBox" – określa układ współrzędnych dla atrybutów x, y, width oraz height:

- **userSpaceOnUse** – x1, y1, width, height - reprezentują współrzędne w lokalnym układzie współrzędnych, dla miejsca wstawiania wzorca;
- **objectBoundingBox –** x1, y1, width, height - reprezentują współrzędne w układzie współrzędnych obiektu do którego wzorzec jest wstawiany;

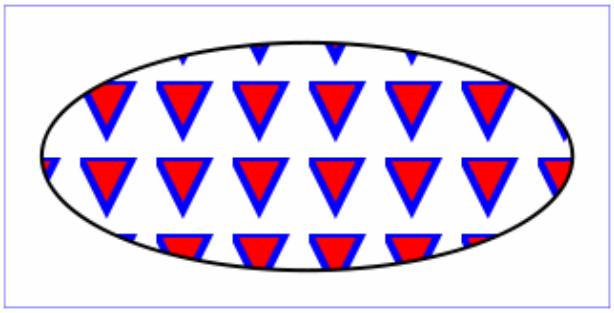
wartością domyślną jest " objectBoundingBox ";

- patternContentUnits = "userSpaceOnUse | objectBoundingBox" – określa układ współrzędnych dla zawartości obiektu pattern. Ignorowane, jeżeli atrybut viewBox został zdefiniowany:
  - **userSpaceOnUse** – x1, y1, width, height - reprezentują współrzędne w lokalnym układzie współrzędnych, dla miejsca wstawiania wzorca;
  - **objectBoundingBox –** x1, y1, width, height - reprezentują współrzędne w układzie współrzędnych obiektu do którego wzorzec jest wstawiany.

  wartością domyślną jest "userSpaceOnUse";
- patternTransform = "<transform-list>" – zob. atrybut transform;
- x = "<coordinate>" – x, y, width oraz height opisują położenie i wielkość pojedynczego elementu wzorca. Wartością domyślną jest "0";
- y = "<coordinate>" – zob. x. Wartością domyślną jest "0";
- width = "<length>" – zob. x. Wartość ujemna jest błędna. Wartość zero wyłącza rysowanie obiektu. Wartością domyślną jest "0";
- height = "<length>" – zob. x. Wartość ujemna jest błędna. Wartość zero wyłącza rysowanie obiektu. Wartością domyślną jest "0";
- xlink:href = "<uri>" – Oonośnik URI do innego wzorca zdefiniowanego w tym samym pliku SVG. Wszystkie argumenty nie zdefiniowane w aktualnym elemencie będą dziedziczone z tego wzorca.

Listing 2.25. Przykład użycia elementu pattern. [Źródło: http://www.w3.org].

```
<?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="8cm" height="4cm" viewBox="0 0 800 400"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<defs>
<pattern id="TrianglePattern" patternUnits="userSpaceOnUse"
x="0" y="0" width="100" height="100"
viewBox="0 0 10 10" >
<path d="M 0 0 L 7 0 L 3.5 7 z" fill="red" stroke="blue" />
</pattern>
</defs>
<rect fill="none" stroke="blue"
x="1" y="1" width="798" height="398"/>
<ellipse fill="url(#TrianglePattern)" stroke="black" stroke-width="5"
cx="400" cy="200" rx="350" ry="150" />
</svg>
```



Rys. 2.28. Przykład użycia elementu pattern. [Źródło: http://www.w3.org].

2.9. Wycinanie ścieżek i nakładanie masek

2.9.1 Ścieżki wycinające

Ścieżki wycinające (clipping paths) – jest to dowolna kombinacja ścieżek (path), napisów (text) oraz kształtów podstawowych. Obcinanie polega na rysowaniu części wspólnej rysowanego obiektu i ścieżki wycinającej.

Dostępne atrybuty:

- clipPathUnits = "userSpaceOnUse | objectBoundingBox" – określa układ współrzędnych dla zawartości clipPath:

- **userSpaceOnUse** – określa lokalny układ współrzędnych, dla miejsca wstawienia clipPath;
- **objectBoundingBox –** określa układ współrzędnych obiektu do którego clipPath jest wstawiana;

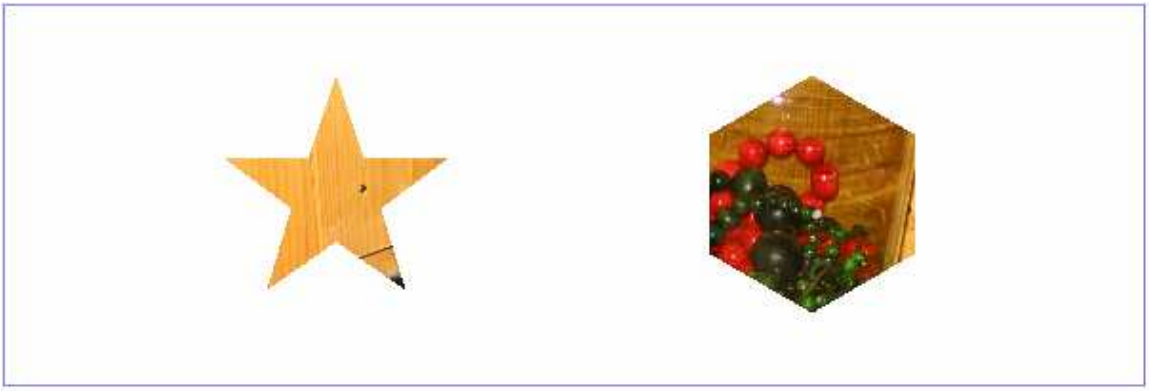
wartością domyślną jest "userSpaceOnUse".

Właściwości:

- clip-path = "<uri> | none | inherit" – określa ścieżkę wycinającą:
  - <uri> – odnośnik do elementu graficznego wewnątrz tego samego dokumentu, który zostanie użyty jako ścieżka wycinająca;
  - inherit – ścieżka wycinająca jest dziedziczona z elementu nadrzędnego;
  - none – jest to wartość domyślna, która nic nie zmienia;
- 'clip-rule' = "nonzero | evenodd | inherit" – właściwość oznacza, w jaki sposób określane jest, czy punkt jest wewnątrz figury, czy nie. Prowadzona jest dowolna prosta przechodząca przez dany punkt i zliczana jest ilość przecięć z daną figurą:
  - nonzero – jeżeli ilość przecięć jest różna od zera to punk należy do figury;
  - evenodd – jeżeli ilość przecięć jest nieparzysta to punk należy do figury;
  - inherit – wartość jest dziedziczona z elementu nadrzędnego.

Listing 2.26. Przykład użycia ścieżki wycinającej.[Źródło: opracowanie własne]

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="800" height="300" viewBox="0 0 1200 600"
preserveAspectRatio = "xMinYMin" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<clipPath id="clip" >
<polygon points="850,75 958,137.5 958,262.5 850,325 742,262.6 742,137.5" />
<polygon points="350,75 379,161 469,161 397,215 423,301 350,250 277,301
303,215 231,161 321,161" />
</clipPath>
</defs>
<rect x="1" y="1" width="1198" height="400" fill="none" stroke="blue"
stroke-width="1"/>
<image x="0" y="0" width="1200" height="600" xlink:href="kot.jpg"
clip-path="url(#clip)" />
</svg>
```



Rys. 2.29. Przykład użycia ścieżki wycinającej. [Źródło Opracowanie własne].

2.9.2. Maski

Maska (mask) – jest to półprzezroczysty element graficzny, lub grupa elementów, które są nakładane na wyświetlaną grafikę.

Dostępne atrybuty:

- maskUnits = "userSpaceOnUse | objectBoundingBox" – określa układ współrzędnych dla atrybutów x, y, width oraz height:
  - **userSpaceOnUse** – x1, y1, width, height reprezentują współrzędne w lokalnym układzie współrzędnych, dla miejsca wstawiania maski;
  - **objectBoundingBox –** x1, y1, width, height reprezentują współrzędne w układzie współrzędnych obiektu do którego maska jest wstawiana;

  wartością domyślną jest " objectBoundingBox "
- maskContentUnits = "userSpaceOnUse | objectBoundingBox" – określa układ współrzędnych dla zawartości obiektu pattern. Ignorowane, jeżeli atrybut viewBox został zdefiniowany:
  - **userSpaceOnUse** – x1, y1, width, height reprezentują współrzędne w lokalnym układzie współrzędnych, dla miejsca wstawiania maski;
  - **objectBoundingBox –** x1, y1, width, height reprezentują współrzędne w układzie współrzędnych obiektu do którego maska jest wstawiana,

  Wartością domyślną jest "userSpaceOnUse"
- x = "<coordinate>" – x, y, width, height określają położenie i rozmiar maksymalnego prostokąta jaki zostanie użyty. Jeżeli element zdefiniowany, jako maska wychodzi poza obszar tego prostokąta, to zostanie obcięty. Jeżeli element mieści się w prostokącie nic się nie stanie. Wartość domyślna dla x wynosi "-10%";

- y = "<coordinate>" – zob. x. Wartość domyślna wynosi "-10%";
- width = "<length>" – zob. x. Wartość domyślna wynosi "120%";
- height = "<length>" – zob. x. Wartość domyślna wynosi "120%".

Właściwości:

- mask = "<uri> | none | inherit " – definiuje maskę:
    - <uri> – odnośnik do elementu graficznego wewnątrz tego samego dokumentu, który zostanie użyty jako maska;
    - inherit – ścieżka wycinająca jest dziedziczona z elementu nadrzędnego;
    - none – jest to wartość domyślna, która nic nie zmienia.

Listing 2.27. Przykład użycia maski. [Źródło: http://www.w3.org].

```
<?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="8cm" height="3cm" viewBox="0 0 800 300" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
<linearGradient id="Gradient" gradientUnits="userSpaceOnUse"
x1="0" y1="0" x2="800" y2="0">
<stop offset="0" stop-color="white" stop-opacity="0" />
<stop offset="1" stop-color="white" stop-opacity="1" />
</linearGradient>
<mask id="Mask" maskUnits="userSpaceOnUse"
x="0" y="0" width="800" height="300">
<rect x="0" y="0" width="800" height="300" fill="url(#Gradient)" />
</mask>
<text id="Text" x="400" y="200"
font-family="Verdana" font-size="100" text-anchor="middle" >
Masked text
</text>
</defs>
<rect x="0" y="0" width="800" height="300" fill="#FF8080" />
<use xlink:href="#Text" fill="blue" mask="url(#Mask)" />
<use xlink:href="#Text" fill="none" stroke="black" stroke-width="2" />
</svg>
```

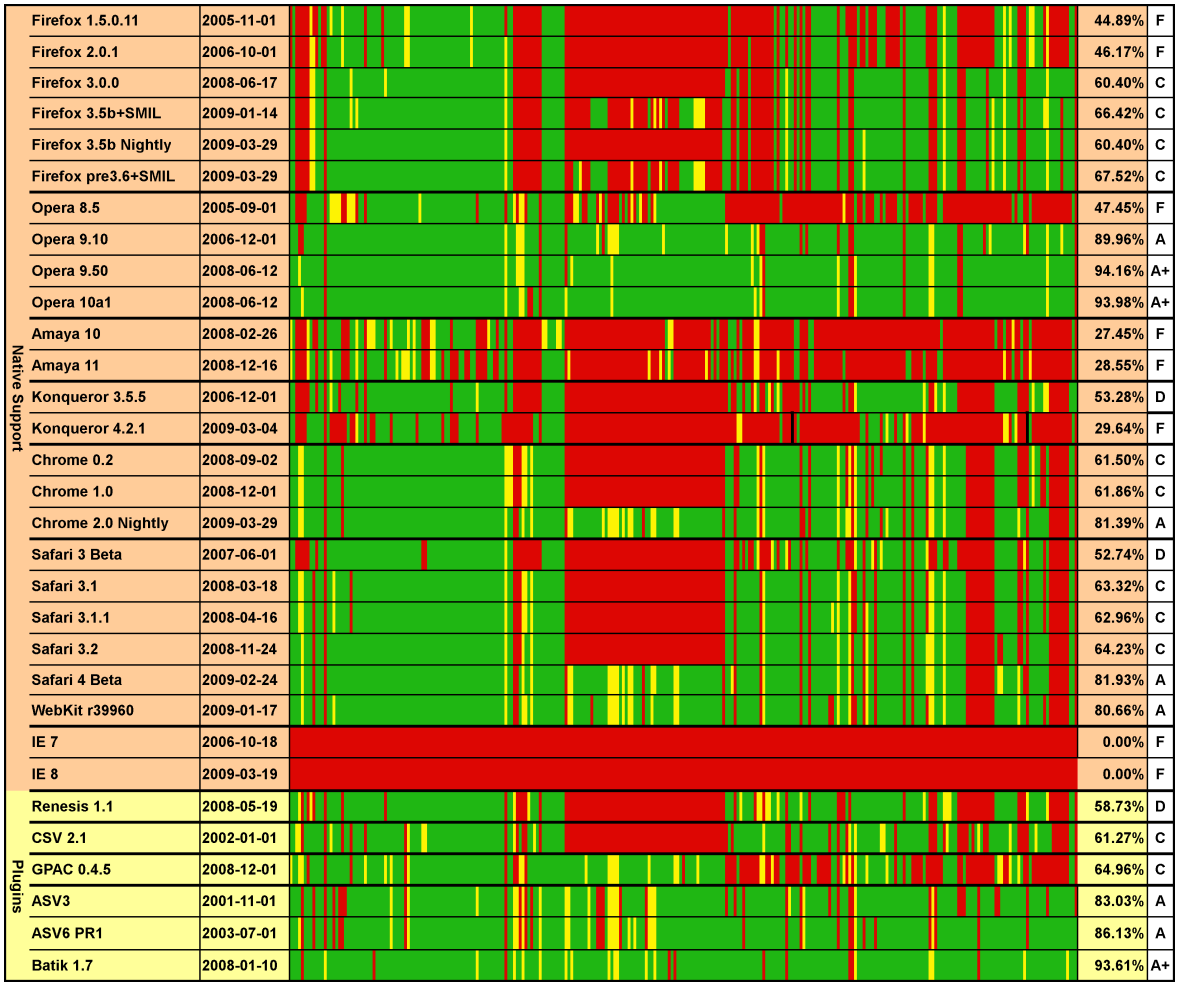
Rys. 2.30. Przykład użycia maski. [Źródło: http://www.w3.org].

3. Wykorzystanie SVG

3.1. Przeglądarki

Format SVG projektowany był z myślą u użyciu go w internecie. Żeby jednak mógł w nim zaistnieć muszą istnieć narzędzia, czyli przeglądarki, które będą poprawnie wyświetlać dokumenty w nim zapisane.

Organizacja W3C udostępnia zestaw ponad trzystu testów (pliki zapisane w formacie SVG i PNG), dzięki którym można sprawdzić czy program wyświetla dokument SVG tak jak powinien on wyglądać.



Rys. 3.1. Obsługa formatu SVG przez programy. [Źródło: http://codedread.com].

Powyższe zestawienie prezentuje wyniki tych testów dla poszczególnych programów. Kolor czerwony oznacza, że test nie został zaliczony, kolor żółty, że wystąpiły niewielkie niezgodności, natomiast kolor zielony oznacza zaliczenie testu. Zestawienie pochodzi z marca 2009 roku i pokazuje, że nie ma jeszcze narzędzia, które by w pełni potrafiło obsłużyć format SVG. Jednak największym problemem jest to, że Internet Explorer – najpopularniejsza przeglądarka, której używa około 70% użytkowników Internetu[10] w ogóle nie wspiera tego formatu. Istnieją oczywiście wtyczki (GPAC, Adobe Viewer), które rozszerzają możliwości Internet Explorera o obsługę formatu SVG, jednak moim zdaniem jest mało prawdopodobne, żeby przeciętny użytkownik instalował takie dodatki. Po pierwsze, jeżeli użytkownik pracuje na komputerze w pracy lub w szkole może się okazać, że nie ma uprawnień żeby zainstalować jakikolwiek program. Po drugie, jeżeli strona nie ładuje się poprawnie, użytkownik najprawdopodobniej ją opuści, a nie będzie tracił czas na wyszukiwanie rozwiązania.

Brak wsparcia dla SVG przez Internet Explorer powoduje praktycznie zablokowanie tego formatu dla większości użytkowników Internetu, a co za tym idzie zablokowanie rozwoju popularności SVG. W ciągu dziesięciu lat istnienia SVG nie zdobył sobie takiej popularności jak np Adobe Flash[11] i trudno ocenić czy uda mu się jeszcze zdobyć szeroka popularność. Moim zdaniem jeżeli do Interent Explorera nie zostanie dodana obsługa SVG (nic nie wiadomo o takich planach) to format ten nie zdobędzie szerokiej popularności.

3.2. Narzędzia

3.2.1. Inkscape

Inkscape[12] jest darmowym programem udostępnianym na zasadach licencji GNU GPL, umożliwiającym wyświetlanie i edycję obrazów zarówno w formatach wektorowych, jak i rastrowych .

SVG jest natywnym formatem programu Inkscape. Jednak podczas zapisywania Inkscape dodaje pewne informacje, ułatwiające późniejszą edycje pliku. Dane te nie wpływają na wygląd obrazu i nie mogą powodować jakichkolwiek problemów

[10] *http://www.networld.pl*
[11] *http://www.flashzone.pl/item/1013/Popularnosc-Flash-Player-6-83-8/*
[12] *http://www.inkscape.org/*

w programach zgodnych z specyfikacją SVG[13]. Żeby uniknąć jakichkolwiek problemów z programami, które nie są zgodne ze specyfikacją SVG, lub jeżeli nie planujemy jeż dalszej edycji pliku, można te informacje usuną zapisując plik w „czystym" SVG.

Inkscape oferuje również możliwość zapisu w formacie skompresowanego SVG – SVGZ, daje nam to znaczne zmniejszenie objętości pliku.

Program ma wbudowany import większości formatów rastrowych (JPG, PNG, GIF i in.), ale eksportować do rastra może tylko w formacie PNG. Przy pomocy rozszerzeń, Inkscape może otwierać i zapisywać formaty PDF, EPS, AI, Dia, Sketch

Inkscape w większości dobrze radzi sobie ze specyfikacją SVG, jednak nie wszystko jest jeszcze obsługiwane. Główne części SVG, nie wspierane jeszcze w Inkscape, to filtry (zawyjątkiem rozmycia Gaussa, obecnego od wersji 0.45), animacja i czcionki SVG.

Inkscape umożliwia również wektoryzację obrazów rastrowych. Wykorzystywany jest mechanizm wektoryzacji bitmap Potrace[14], który umożliwia trzy opcje filtrowania[15]:

- Brightness cutoff (Odcinanie jasności) – Filtr używa tylko sumy czerwieni, zieleni i błękitu (lub odcieni szarości) piksela do wskazania, czy będzie uważany za czarny, czy biały. Próg może być ustawiony od 0,0 (czerń) do 1,0 (biel). Im wyższe ustawienie, tym ciemniejszy obraz pośredni.
- Edge Detection (Wykrywanie krawędzi) – używa algorytmu wymyślonego przez J.Canny'ego, dla szybkiego znajdowania ciągłych linii o podobnym kontraście. Tworzy pośrednią bitmapę o wyglądzie mniej podobnym do oryginalnego obrazu niż rezultaty Progu Brightness (Jasność), ale prawdopodobnie dostarczy informacji o krzywych, które inaczej zostałyby zignorowane.
- Color Quantization (Kwantowanie koloru) – wynikiem działania tego filtra jest tymczasowy obraz, bardzo różny od obu poprzednich, ale naprawdę bardzo użyteczny. Zamiast pokazywania obrysów jednakowych jasności czy kontrastów, znajduje krawędzie, na których zmieniają się kolory, nawet przy równej jasności i kontraście.

13 *http://www.usinginkscape.com/*
14 *http://potrace.sourceforge.net/*
15 *http://www.inkscape.org/doc/*

3.2.2. GeoTools

GeoTools[16] jest darmową biblioteką języka Java udostępnianą na zasadzie licencji LGPL. Został on stworzony, aby wspomagać programistów w pisaniu aplikacji typu GIS. Biblioteka ta dostarcza gotowe implementacje funkcji pozwalających na gromadzenie, przetwarzanie i prezentację danych przestrzennych zgodnie ze standardami OGC (Open Geospatial Consortium). GeoTools udostępniany jest w formie gotowych archiwów Java (JAR files), w których zaimplementowane są klasy języka Java. Oprócz plików typu JAR, istnieje także możliwość dołączania pełnych kodów klas, które wchodzą w skład pakietu GeoTools.

GeoTools jest w praktyce używany w wielu projektach programistycznych o charakterze naukowym i komercyjnym. Implementuje specyfikacje OGC (Open Geospatial Consortium), ponieważ jest rozwijana we współpracy z GeoAPI (zbiór interfejsów dla danych geoprzestrzennych).

GeoTools obsługuje wiele formatów danych np.:

- SVG (zapis);
- ESRI Shapefile (odczyt/zapis);
- GML – Geography Markup Language (odczyt);
- WFS – OGC Web Feature Server (odczyt/zapis);
- WMS – OGC Web Mapping Server client (odczyt);
- PostGIS – rozszerzenie PostgreSQL dla danych geoprzestrzennych (odczyt);
- Oracle Spatial – rozszerzenie Oracle dla danych geoprzestrzennych (odczyt);
- MySQL – rozszerzenie MySQL dla danych geoprzestrzennych (odczyt);
- ESRI ArcSDE (odczyt);
- GeoMedia (odczyt);
- Tiger (odczyt);
- ArcGrid (odczyt/zapis);
- GeoTIFF (odczyt).

GeoTools jest wykorzystywane przez wiele projektów np:

- GeoServer;
- GeoVISTA Studio;

[16] *http://geotools.codehaus.org*

- MyMaps;
- The Balloon Project;
- uDig.

3.2.3. Batik

Batik[17] jest to napisany w języku Java zbiór narzędzi dla aplikacji lub apletów mających na celu obsługę formatu SVG. Ponieważ Batik oparty jest na technologii Java, może być wykorzystany wszędzie tam gdzie Java jest zainstalowana (komputery osobiste, palmtopy, telefony komórkowe)

Batik można podzielić na dwie grupy:

- biblioteka języka Java, która zawiera następujące moduły:
    - implementację modelu DOM (Dokument Object Model);
    - zbiór parserów języka SVG;
    - moduł obsługi skryptów;
    - generator, który tworzy dokument SVG na podstawie odwołań do Java2D;
    - komponenty SVG z biblioteki swing;
    - transcoder umożliwiający rasteryzację;
- zbiór narzędzi i aplikacji:
    - squiggle – przeglądarka SVG;
    - rasteryzer obrazów SVG;
    - konwerter fontów TTF (Tru Type Font) do SVG;
    - pretty printer – drukarka dokumentów SVG.

Przeglądarka Batik jest najlepszym narzędziem do wyświetlania dokumentów SVG pod względem zgodności ze specyfikacją SVG.

[17] *http://xmlgraphics.apache.org/batik*

3.3. Aplikacje typu serwer

3.3.1. MapServer

MapServer[18] jest projektem open source przeznaczonym do tworzenia aplikacji internetowych obsługujących dane geograficzne. MapServer oferuje szerokie możliwości np.:

- obsługa formatów wektorowych:
  - SVG;
  - ESRI shape;
  - PostGIS;
  - ESRI ArcSDE i inne;
- obsługa formatów rastrowych:
  - TIFF/ geoTIFF;
  - EPPL7;
  - pozostałe obsługiwane przez GDAL (Geospatial Data Abstraction Library);
- indeksowanie przestrzennych drzew quadtree dla plików shape;
- możliwość swobodnego dostosowania danych wyjściowych do potrzeb użytkownika poprzez szablony;
- obsługa formatu TrueType;
- obsługa danych rastrowych i wektorowych;
- możliwość tworzenia elementów map takich jak legenda, mapa wzorcowa, skala;
- zależna od skali wizualizacja elementów map;
- tworzenie map tematycznych przy wykorzystaniu podstawowych klas wyrażeń logicznych i wyrażeń regularnych;
- możliwość opisu elementów mapy z wykrywaniem kolizji;
- bezpośrednie konfigurowanie poprzez Internet;
- bezpośrednie rzutowanie.

[18] *http://mapserver.org/*

Oprogramowanie MapServer zawiera system MapScript umożliwiający wykorzystywanie języków skryptowych PHP, Perl, Python.

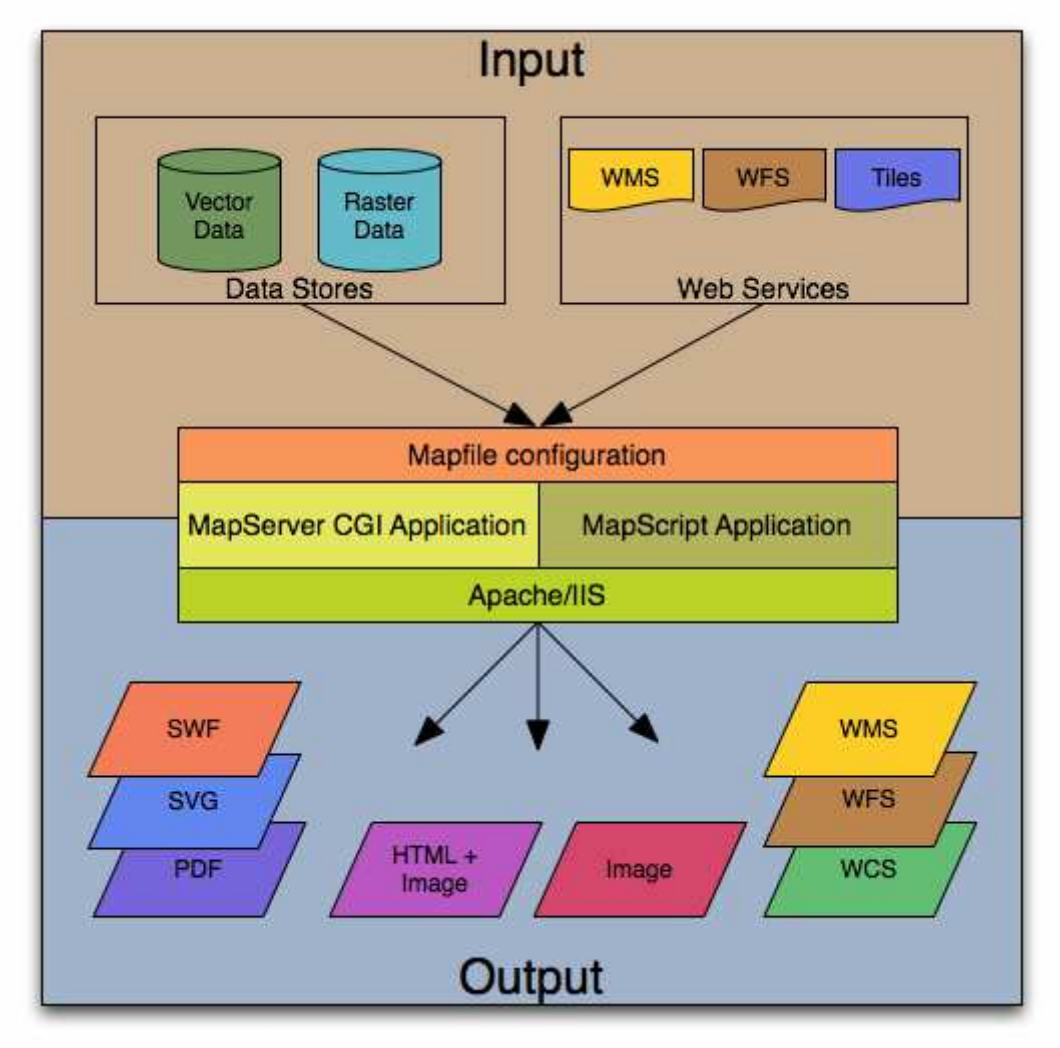
MapScript zapewnia rozbudowane środowisko do tworzenia aplikacji wykorzystujących i integrujących dane w różnych formatach. Jeżeli dane zawierają składowe przestrzenne, do których dostęp jest możliwy poprzez środowisko języka skryptowego, to możliwe jest ich zobrazowanie w postaci mapy.

Przykładowo, poprzez wykorzystanie modułu Perl DBI, możliwe jest zintegrowanie danych niemalże z dowolnej bazy danych (np. Oracle, Sybase, MySQL), z danymi GIS w pojedynczym odwzorowaniu w postaci mapy, na przykład na stronie internetowej.

Ponadto MapServer obsługuje szereg specyfikacji internetowych ustalonych przez Open Geospatial Consortium. W chwili obecnej oprogramowanie MapServer wspiera:

- WMS (Web Map Service, klient/serwer);
- WFS (Web Feature Service, klient/serwer);
- WMC (Web Map Context);
- WCS (Web Coverage Service).

Poniżej przedstawiam schemat budowy aplikacji MapServer:



Rys. 3.2. Architektura aplikacji MapSerwer. [Źródło: http://mapserver.org].

- plik mapy (map file) – jest to konfiguracyjny plik tekstowy dla aplikacji MapServer. Definiuje obszar wyświetlanej mapy, źródła danych, warstwy, definiuje również, gdzie mapa ma zostać wyświetlona itp.;
- dane geograficzne – MapServer może wykorzystywać dane geograficzne ze źródeł różnego typu. Formatem podstawowym jest ESRI shapefile;
- strona HTML – interfejs pomiędzy użytkownikiem, a aplikacją MapServer. Program CGI (Common Gateway Interface) umożliwia dynamiczne przetwarzanie obrazów oraz danych. Programy CGI są bezstanowe, co oznacza, że nie pamiętane są żadne poprzednie zapytania. Dlatego też, jeżeli aplikacja wykonuje zapytanie do MapServera, za każdym razem musi wysłać informację o zawartości (używane warstwy, aktualna pozycja na mapie itp.) Prosta aplikacja CGI MapServer może zawierać dwie strony html:

- plik inicjalizacyjny – używany do wysłania początkowego zapytania do serwera http i MapServer;
- szablon – opisuje, jak mapy i legendy, wygenerowane przez MapServer, będą wyświetlane w przeglądarce. Może również opisywać, w jaki sposób użytkownik może współdziałać z aplikacją MapServer (przeglądanie, skalowanie, wysyłanie zapytań);

➢ MapServer CGI – plik binarny lub wykonywalny, który otrzymuje zapytania i zwraca dane;

➢ serwer HTTP z zainstalowaną aplikacją MapServer.

MapServer umożliwia zapis do formatu SVG. Żeby to zrobić należy w pliku mapfile ustawić parametr IMAGETYPE na svg, zdefiniować obiekt OUTPUTFORMAT, oraz w obiekcie WEB zdefiniować parametry IMAGEPATH i IMAGEURL (listing 3.1).

Listing 3.1. Konfiguracja pliku mapfile dla programu MapServer [Źródło: http://mapserver.org].

```
MAP
        IMAGETYPE svg
                OUTPUTFORMAT
                        NAME svg
                        MIMETYPE "image/svg+xml"
                        DRIVER svg
                        FORMATOPTION  "COMPRESSED_OUTPUT=TRUE"
                        FORMATOPTION  "FULL_RESOLUTION=TRUE"
                END
                WEB
                        IMAGEPATH "/tmp/ms_tmp/"
                        IMAGEURL  "/ms_tmp/"
                END
        END
END
```

Opcja COMPRESSED_OUTPUT ustawiona na TRUE umożliwia zapis skompresowanego SVG, czyli SVGZ.

Żeby przetestować działanie zapisu do formatu SVG można użyć Map Server CGI np.

wpisując przykładowy adres URL:

Listing 3.2. Generowanie SVG za pomacą MapServer poprzez zapytanie URL [Źródło: http://mapserver.org].

```
http://127.0.0.1/cgi-bin/mapserv.exe?map=my/path/to/my-svg.map&mode=map&layers=layer1 layer2
```

lub korzystając z PHP/MapScript - plik php powinien wyglądać tak:

Listing 3.3. Generowania SVG za pomocą MapServer poprzez skrypt php [Źródło: http://mapserver.org].

```
<?php
        dl("php_mapscript_45.dll");
        $oMap = ms_newmapObj("my/path/to/my-svg.map");
        $img = $oMap->draw();
        header("Content-type: image/svg+xml");
        $url = $img->saveImage("");
?>
```

Plik SVG zostanie utworzony w folderze IMAGEPATH.

3.3.2. GeoServer

Geoserver[19] został stworzony w roku 2001 przez The Open Planning Project (TOPP). Jest oprogramowaniem opartym na licencji open source napisanym w Javie. Pozwala użytkownikowi na udostępnienie i edycję danych geoprzestrzennych.

Będąc projektem społecznościowym, geoserver jest rozwijany, testowany i wspierany przez różnego rodzaju grupy. Georerver stosuje się do standardów wyznaczonych przez Open Geospatial Consortium (OGC) – Web Feature Service (WFS) i Web Coverage Service (WCS).

[19] *http://geoserver.org/*

Zaletą GeoServer jest to, że szybko dostosowywany jest do nowych technologii. GeoServer był jednym z pierwszych produktów, które obsługiwały format KML (Google Earth, Gogle Maps). Podobnie było z tile renderingiem (dzielenie obrazu na kawałki), zarekomendowanym przez Web Map Service.

Geoserver napisany jest w Java, dlatego też może w łatwy sposób współpracować z innymi projektami napisanymi w Java. Większość zadań związanych z wczytywaniem i zapisywaniem do różnego rodzaju formatów, transformacją układów współrzędnych i renderowaniem, obsługiwana jest przez biblioteki.

Powiązanie z innymi projektami, open source nie jest ograniczone tylko do płaszczyzny geoprzestrzennej. Geoserver korzysta z wielu innych bibliotek. Najbardziej znaną jest Spring Framework, w pełni funkcjonalny framework („struktura wspomagająca tworzenie, rozwój i testowanie powstającej aplikacji”[20]) przeznaczony do budowania aplikacji w języku Java. Przez ostatnie lata Spring stał się najczęściej używanym szablonem. Jego przewagą nad J2EE jest prostota i stosunkowo niskie nakłady czasu pracy przy tworzeniu małych i średnich aplikacji. Jedną z ważniejszych zalet korzystania z biblioteki Spring, jest możliwość skorzystania z systemu bezpieczeństwa Acegi. Acegi wspiera różnego rodzaju autentykacje, usługi katalogowe (LDAP) i inne.

Ważną biblioteka wykorzystywaną przez GeoServer jest Freemarker. Jest to tzw. Template engine, czyli „system pozwalający na rozdzielenie warstwy pozyskiwania i obróbki danych od warstwy ich prezencji”.[21] Umożliwia on użytkownikowi dopasowanie wyświetlanych danych do własnych potrzeb.

Geoserver WMS wspiera SVG jako format wynikowy. Żeby serwer zwrócił SVG należy parametr Format ustawić na image/svg+xml. Oto przykład takiego zapytania:

---

[20] *http://pl.wikipedia.org/wiki/Framework*

[21] *http://binboy.sphere.pl/index.php?show=132*

Listing 3.4. Generowanie SVG za pomocą Geoserver poprzez zapytanie URL [Źródło: http://geoserver.org].

```
http://localhost:8080/geoserver/wms?
bbox=-130,24,-66,50&
request=GetMap&
layers=states&
width=800&
height=400&
srs=EPSG:4326&
styles=population&
format=image/svg+xml
```

Geoserver umożliwia dwa sposoby renderowania SVG – Simple i Batik. Simple jest szybki, jednak w ograniczony sposób odzwierciedla stylizację SLD (Styled Layer Descriptor). Batik jest wolny, jednak w pełni odzwierciedla stylizację SLD. Sposób renderowania, jaki ma zostać użyty można określić w zakładce Config->WMS ->Rendering.

3.3.3. Google Maps

Google Maps[22] – jeden z serwisów firmy Google, który z założenia ma być bazą danych map oraz zdjęć satelitarnych i lotniczych całego świata. Zdjęcia oglądać można w kilkunastu skalach, jednak poziom największego zbliżenia jest różny w różnych miejscach.

Geokoder Google Maps parsuje adresy wpisane w języku naturalnym. Usługa ta działa w większości państw europejskich, gdzie w największych miastach możliwe jest wyszukiwanie pojedynczych domów. Serwis wskazuje również połączenia drogowe między znalezionymi miejscami z przybliżonym czasem podróży i w rozbiciu na etapy. Przebieg trasy może zostać dowolnie zmodyfikowany. Możliwe jest także wyszukiwanie miejsc za pomocą współrzędnych geograficznych (stopni, opcjonalnie minut oraz sekund), na przykład wpisanie 52 13 54 N 21 00 20 E spowoduje znalezienie Pałacu Kultury i Nauki w Warszawie.

[22] *http://maps.google.pl/*

W maju 2007 do Google Maps dodano Street View – funkcja, która zapewnia 360° panoramiczne widoki z poziomu ulicy i pozwala użytkownikom na wyświetlanie wybranych części miasta. Google Street View wyświetla zdjęcia, które wcześniej zostały wykonane z poziomu ulicy przez kamery zamontowane na samochodzie. Do poruszania się używa się klawiszy strzałek na klawiaturze i myszy, aby zmienić kierunek i kąt. Wzdłuż ulic wyświetlane są linie pomocnicze wskazujące możliwy kierunek dalszego poruszania.
W maju 2009 r. miała zostać sfotografowana Warszawa, a zaraz po niej Kraków.

Google stworzyło API, umożliwiające wstawienie własnej mapy na dowolną stronę internetową. Dostęp do API odbywa się z poziomu języka JavaScript, ActionScript 3 (Google Maps API for Flash), lub w postaci obrazu (Google Static Maps API). Do korzystania z Google Maps API wymagany jest bezpłatny klucz, który może uzyskać każdy użytkownik konta Google. Klucz pozwala na dostęp z jednej domeny lub katalogu domeny. Google Maps API umożliwia zintegrowanie ze stroną internetową w pełni funkcjonalnej mapy, z własnymi danymi i funkcjami do obsługi zdarzeń.
Pierwsze wersje API nie oferowały zaawansowanych funkcji, dostępnych w witrynie Google Maps. Najnowsza wersja 2.8 umożliwia m.in:

- możliwość geokodowania adresów;
- rysowanie polilinii;
- wyznaczanie tras przejazdu wraz z listą kroków;
- pełna kontrola widoku ulic (Street View);
- wsparcie dla języka KML/GeoRSS.

Google Maps od wersji 2 (koniec 2007r.) używa SVG do rysowania polilini. Kod SVG generowany jest przez bibliotekę Google Web Toolkit napisaną w JavaScript. Kod SVG generowany jest tylko dla przeglądarek, które potrafią go wyświetlić (np. Firefox, Opera, Safari), dla Internet Explorera generowany jest kod VML (Vector Markup Language – język z rodziny XML służący do generowania grafiki wektorowej stworzony przez Microsoft).

Podstawowym problemem systemów wizualizacji jest fakt, że serwer posiada przeważnie więcej danych niż użytkownik chce zobaczyć. Przepustowość łącza i moc obliczeniowa nie są wystarczające żeby przetworzyć i przesłać takie ilości danych. Dlatego konieczne jest podzielenie danych po stronie serwera i wysłanie do klienta tylko tych

najistotniejszych. Jeżeli użytkownik zmieni obszar wyświetlanej mapy lub przybliżenie dodatkowe dane będą musiały być załadowane z serwera.

Google Maps, poprzez Java Script dynamicznie modyfikuje zawartość SVG, który ma być wyświetlany przez przeglądarkę. Algorytmy obsługi danych geograficznych są prostsze, ponieważ końcowe renderowanie grafiki wykonywane jest przez przeglądarkę. Dodatkową zaletą jest fakt, że rozwiązanie to jest niezależne od platformy sprzętowej[23].

3.4. Aplikacje typu desktop

3.4.1. gvSIG

gvSIG[24] jest programem przeznaczonym do zarządzania danymi geoprzestrzennymi. Charakteryzuje się przyjaznym dla użytkownika interfejsem, umożliwiającym szybki dostęp do najczęściej używanych formatów wektorowych i rastrowych. Umożliwia też łączenie się z serwerami: WMS (Web Map Service), WCS (Web Coverage Service), WFS (Web Feature Service) .

Głównymi zaletami gvSig są:

- przenośność – jest napisany w Javie, więc jest dostępny wszędzie tam gdzie Java jest zainstalowana;
- modułowość – umożliwia łatwy dalszy rozwój oprogramowania;
- otwartość – udostępniany na zasadach licencji GPL.

gvSig jest zbudowany z trzech odrębnych części:

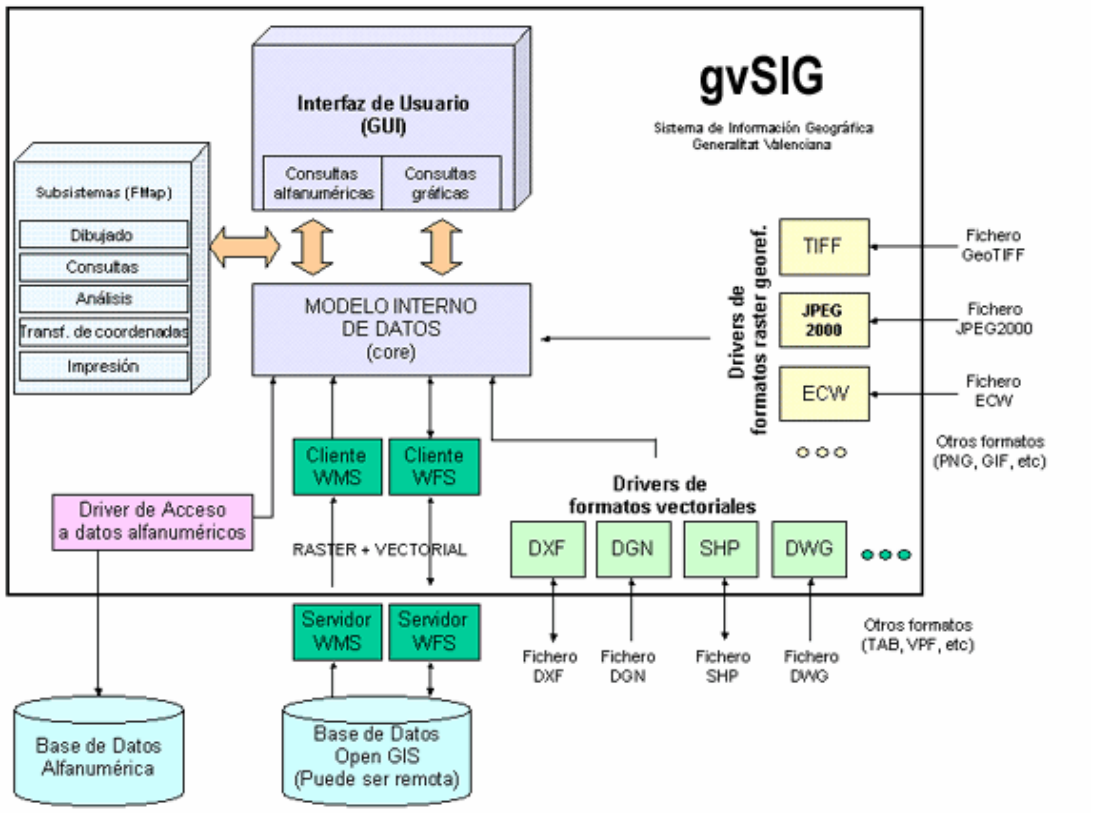
- **ANDAMI**: – główna aplikacja z możliwością rozszerzania funkcjonalności poprzez instalację wtyczek (plugin'ów). Moduł odpowiedzialny jest za start aplikacji, inicjalizację okien, obsługę rozszerzeń;
- **FMAP**: - biblioteka klas umożliwiająca tworzenie aplikacji GIS. Odpowiedzialna jest za odczyt i zapis obsługiwanych formatów, rysowanie map, przypisywanie legendy, definiowanie symboli, obsługę

23 *http://www.pms.ifi.lmu.de/publikationen/PMS-FB/PMS-FB-2006-5/otn2svg.pdf*

24 *http://www.gvsig.gva.es/*

zapytań (wyszukiwanie, analiza). Zawiera jądro (core) definiujące wewnętrzny system danych;

- **GUI**: – interfejs umożliwiający komunikację użytkownika z programem.

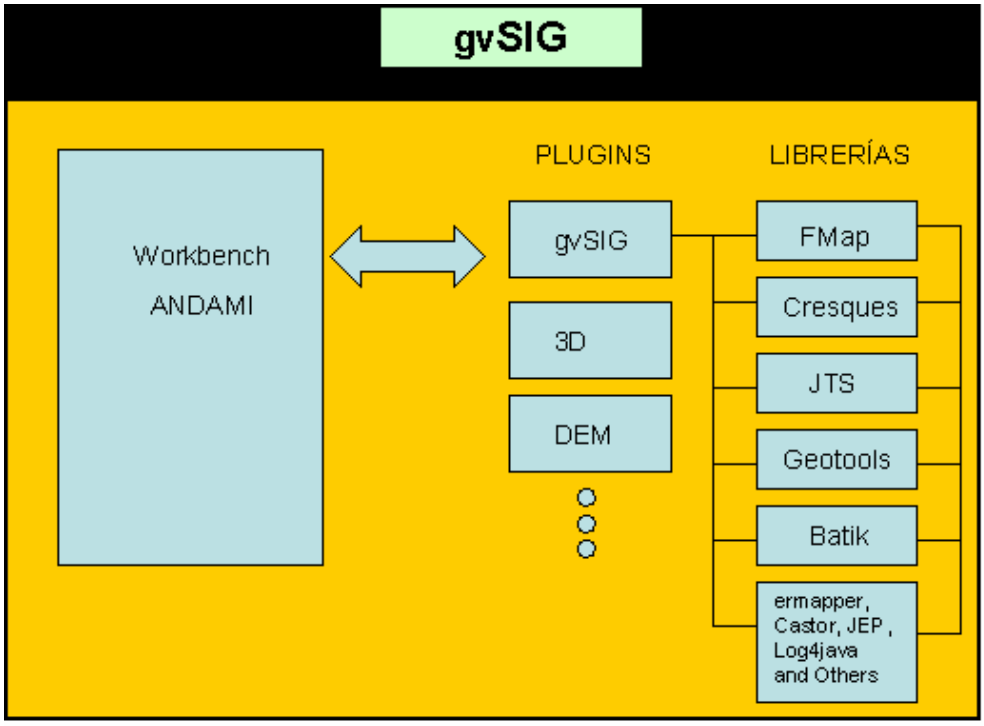


Rys. 3.3. Schemat działania aplikacji gvSig.[Źródło: http://www.gvsig.gva.es].

Schemat budowy aplikacji:

- sterowniki (drivers) odpowiedzialne są za odczyt danych z plików (.shp, .dgn, .dxf, .dwg, .ecw, .tiff, .jpeg,), oraz serwerów (WMS OpenGis (Web Map Server), WFS (Web Feature Server)), jak i zapis danych (.shp, .dxf);
- budowa struktury obiektów systemu danych jądra;
- moduł FMAP może być podzielony ze względu na funkcjonalność:
  - rysowanie warstw (rastrowych i wektorowych) i przypisywanie odpowiednich symboli;
  - obsługa zapytań;
  - tworzenie analizy;
  - transformacje układów współrzędnych;

- moduł GUI umożliwia interakcje z użytkownikiem, implementuje funkcjonalność dla menu, przycisków, okien dialogowych itp.



Rys. 3.4. Schemat budowy aplikacji gvSig. [Źródło: http://www.gvsig.gva.es].

Program gvSIG obsługuje format SVG, jednak w dość ograniczonym zakresie. Podczas tworzenia mapy możemy dodać do projektu warstwę zapisaną w formacie SVG (przycisk dodaj obraz), jednak jedyne, co można zmienić to ustawienia związane z całością warstwy takie jak położenie, rozmiar i obrót warstwy. Niestety nie można zmieniać zawartości takiego obrazu.

Format SVG wykorzystywany jest jeszcze przy użyciu symboli oznaczających strzałkę północy. Są one zapisane w SVG i przechowywane w folderze „bin\gvSIG\extensiones\com.iver.cit.gvsig\northimages”. Można dodać własny symbol zapisując go w tym folderze, program wczyta go automatycznie.

3.4.2. Quantum Gis

Quantum GIS[25] (QGIS) jest oprogramowaniem geoinformacyjnym (GIS) napisanym w językach C++ i Pyton, dostępnym na zasadach licencji GNU General Public License (GPL).

Program może zostać uruchomiony na różnych platformach systemowych (Linux, UNIX, Mac OS X i Windows). Dzięki systemowi wtyczek (plugin'ów), w prosty sposób można rozszerzać funkcjonalność systemu, w zależności od własnych potrzeb. Quantum Gis może również pełnić rolę, jako graficzny interfejs dla oprogramowania GRASS (Geographic Resources Analysis Support System), co daje dostęp do funkcjonalności tego rozbudowanego programu.

Quantum GIS umożliwia m.in.:

- wyświetlanie i nakładanie danych wektorowych i rastrowych bez konieczności konwersji do wspólnego formatu;
- tworzenie map i przeglądanie danych przestrzennych z wykorzystaniem intuicyjnego graficznego interfejsu użytkownika, który oferuje m.in.:
    - kreator map;
    - zakładki;
    - edycję, podgląd i wyszukiwanie atrybutów;
    - dodanie do mapy siatkę kartograficzną i legendę;
    - obsługę projektu (zapis, przywracanie poprzednich wersji);
- tworzenie, edycję i eksport danych przestrzennych;
- tworzenie analiz danych przestrzennych;
- publikacja map w internecie poprzez eksport do Mapfile (wymaga web serwera z zainstalowaną aplikacją MapServer);
- możliwość dostosowania quantum GIS do własnych potrzeb poprzez system wtyczek (plugin).

Quantum Gis umożliwia eksport projektu do formatu SVG poprzez kompozytor map (menu plik –> asystent wydruku). Kompozytor map umożliwia również wydruk oraz eksport do formatów rastrowych (bmp, png i in.). Kompozytor map składa się z płótna, którego rozmiar możemy dowolnie zdefiniować. Płótno służy do rozmieszczania

[25] *http://www.qgis.org*

poszczególnych elementów, takich jak widok, legenda, skala oraz tekst. Każdy z tych elementów może zostać użyty wiele razy. Najistotniejsze są widoki, dzięki którym można zaprezentować dowolny fragment mapy, w dowolnym powiększeniu. Po skomponowaniu mapy całość można wyeksportować do odpowiedniego formatu.

Quantum Gis wykorzystuje również SVG do prezentacji symboli graficznych. Dla warstw wektorowych, które prezentują dane punktowe, można zdefiniować, jaki symbol w miejscu punktów będzie wyświetlany. Symbole te są zapisane w formacie SVG i przechowywane są w katalogu svg, który znajduje się w katalogu głównym Quantum Gis. Można dodać własne symbole umieszczając je w tym katalogu. Z dodanych symboli można skorzystać dopiero po ponownym uruchomieniu programu.

Podsumowanie i wnioski

Moja praca dotyczy formatu SVG oraz możliwości wykorzystania go w systemach informacji przestrzennej. Ponieważ SVG jest formatem otwartym, stworzonym z myślą o wykorzystaniu w internecie, wydaje się być doskonałym kandydatem na uniwersalny format grafiki wektorowej. Ponieważ jest to format tekstowy dostajemy możliwość przeszukiwania dokumentów po ich zawartości, co jest wielką przewagą SVG nad innymi dostępnymi formatami.

Omawiany przeze mnie format zdobył sobie już pewną popularność wśród użytkowników, a w Internecie można znaleźć coraz więcej obrazów stworzonych w tym właśnie formacie, jednak brak wsparcia przez najpopularniejszą z przeglądarek - Internet Explorer, której używa około 70% użytkowników Internetu[26] powoduje, że format nie może być w łatwy sposób stosowany w Internecie bez pominięcia tych użytkowników.

Ponieważ SVG jest formatem otwartym, umożliwia to tworzenie i rozwój darmowych narzędzi do jego obsługi. Niestety nie ma żadnego narzędzia, które by w pełni poprawnie potrafiło wyświetlić pliki SVG. Twórcy formatu zadbali o to, żeby pojawiła się specyfikacja opisująca wszystkie aspekty formatu, jednak narzędzia do wyświetlania i edycji tworzone są niezależnie od siebie i nie są przez nikogo kontrolowane. Problem polega na tym, że nie jesteśmy w stanie kontrolować sposobu, w jaki ktoś będzie oglądał udostępnione przez nas pliki. Sytuacja może wyglądać tak, jak ze stronami WWW. Każda przeglądarka ma własny silnik odpowiedzialny za wyświetlanie stron i bardzo często występują różnice pomiędzy tym, co wyświetlają poszczególne przeglądarki. Tworząc stronę, niekiedy trzeba poświęcić prawie 90% czasu na dostosowanie strony, żeby wszędzie wyglądała tak samo, czasami niestety jest to jednak niemożliwe.

Przedstawione przeze mnie programy GIS wykorzystują SVG, jednak tylko fragmentarycznie, do specyficznych zastosowań. Najczęściej jest to możliwość wyświetlenia lub zapisania do formatu SVG. Nie znalazłem programów GIS, które by potrafiły edytować dokumenty zapisane w tym formacie. Wnioskuję z tego, że przyszłość

[26] *http://www.networld.pl*

SVG ograniczona jest do szczegółowych zastosowań takich jak: wyświetlanie symboli graficznych niewielkich rozmiarów, lub generowanie dokumentów nie przeznaczonych do dalszej edycji.

Moją pracę podzieliłem na trzy części. W pierwszej starałem się przybliżyć zagadnienia wstępne dotyczące rodzajów grafiki, a także samego formatu SVG. Druga część dotyczy specyfikacji SVG 1.1, opisałem w niej szczegółowo sposoby wykorzystania jej poszczególnych elementów do tworzenia grafiki wektorowej. Trzecia część pracy została poświęcona przedstawieniu różnych systemów informacji przestrzennej, w których można wykorzystać format SVG, oraz innych programów, nie związanych z GIS, jednak dzięki którym możemy wyświetlać i edytować dokumenty SVG.

Pisząc pracę mogłem krytycznie podejść do opisywanego przeze mnie formatu. Lepiej poznać jego zastosowanie przy tworzeniu grafiki wektorowej, co niewątpliwie będę mógł wykorzystać w dalszym rozwoju zawodowym.

Bibliografia

1. Holzner Steven, XML *Vademecum Profesjonalisty*, Wydawnictwo Helion, Gliwice 2001r.
2. Kowal Justyna, *Import i eksport danych hydrometeorologicznych z wykorzystaniem formatu XML oraz interfejsu SAX*, Praca magisterska, Politechnika Krakowska 2008 r.
3. Specyfikacja SVG 1.1 http://www.w3.org/TR/SVG11
4. Strona domowa GeoTools http://geotools.codehaus.org
5. Strona domowa Batik http://xmlgraphics.apache.org/batik
6. Strona Domowa MapServer http://mapserver.org/
7. Strona Domowa Geoserver http://geoserver.org/
8. Strona domowe gvSIG http://www.gvsig.gva.es/
9. Strona Domowa Openmap http://openmap.bbn.com
10. Strona Domowa Quantum Gis http://www.qgis.org
11. Specyfikacja XML http://www.w3.org/XML/
12. Strona domowa Inkscape http://www.inkscape.org/
13. Strona domowa GoogleMaps http://mapy.google.pl/
14. Serwisy webowe, a XML http://serwisy.blogspot.com/2008/07/serwisy-webowe-xml.html
15. Wikipedia *http://en.wikipedia.org*
16. Grafika wektorowa http://www.ikkiz.pl/
17. *Statystyki przeglądarek http://www.networld.pl*
18. Obsługa formatu SVG http://codedread.com

Spis listingów

Spis rysunków