SVG
Scalable Vector Graphics

Emanuel Venturi

Indice

Abstract

SVG è un formato di grafica vettoriale, prodotto dal W3C (World Wide Web Consortium) consorzio no-profit che si occupa degli standard WEB.

A differenza della grafica raster, tipica dei formati bitmap, in cui l'immagine viene codificata come un insieme di pixel, la grafica vettoriale descrive il proprio contenuto per mezzo di figure e curve combinate fra loro.

Il sorgente di un file SVG è puro testo XML™, ed è quindi componibile modularmente con qualsiasi altra applicazione XML.

Tale grammatica descrive le informazioni per mezzo di una struttura dati realizzata ad albero, in cui è possibile identificare nodi padri e nodi figli, partendo da una radice, fino ad arrivare alle foglie. Ogni elemento è racchiuso da tag, e descritto da attributi; un'immagine SVG può essere generata dinamicamente a partire dai dati contenuti in database XML.

Supponiamo di disegnare una bicicletta: di certo appariranno un manubrio, due ruote, due pedali, un sellino e così via. La rappresentazione grafica di ciascuno di questi componenti non è altro che un insieme di comandi di visualizzazione, che il browser o l' SVG-player interpretano, fornendo il risultato finale.

Mentre i tradizionali formati grafici si limitano al rendering, SVG va ben oltre …
Ad ogni componente è possibile associare un insieme di altre informazioni come, ad esempio, un nome, i colori disponibili, il prezzo, o le relazioni con gli altri oggetti.
Sotto questo nuovo punto di vista, un file SVG non è più la semplice rappresentazione grafica di un oggetto, bensì può tranquillamente definirsi una sorta di *database* che descrive informazioni tramite l'uso di una grammatica XML; la sua visualizzazione, quindi, non è altro che una delle infinite possibilità di rappresentarne il contenuto.

Con una semplice trasformazione XSL è possibile cambiare i connotati alla nostra bicicletta, scegliendo di volta in volta i componenti che ci interessano, e visualizzando, oltre che la riproduzione dell'intero assemblato, anche il costo complessivo o i tempi medi di attesa di ogni singolo componente. E con l'uso di CSS (Cascade Style Sheets) la formattazione grafica ed il layout della pagina sono ancora più semplici ed efficienti.

Esistono poi situazioni in cui il rendering grafico di una figura non sempre è possibile, ad esempio quando si lavori con dispositivi mobili a bassa risoluzione e risorse grafiche limitate, oppure quando non vi sia affatto un supporto grafico, ma solo testuale o sonoro; bastano poche righe di XSL per avere una descrizione *testuale* della bicicletta costruita, o addirittura una sua riproduzione tramite *sintetizzatore vocale*.

Aggiungiamo, ora, il fatto che SVG supporta il DOM, ed ogni suo elemento può essere identificato e gestito da un qualunque linguaggio di scripting; un semplice movimento del mouse scatena una serie di eventi che catturati e gestiti opportunamente, innescano elaborazioni di ogni tipo.

… Definireste ancora SVG come un semplice "formato grafico" ? …

Cos'è SVG?

SVG è una grammatica XML per la descrizione di oggetti grafici bidimensionali. Gestisce contemporaneamente vettori, immagini raster, animazione e testo. Come linguaggio XML può essere processato con i tradizionali tool XML come parser, validatori, editor, e browser (SVG è già supportato dalle attuali versioni di MS Internet Explorer e Netscape, anche se è necessario un apposito plug-in).

Per un breve richiamo sulle caratteristiche di XML, si consulti l'APPENDICE B.

SVG definisce un nuovo formato di grafica vettoriale, che fa uso di CSS (Cascade Style Sheets) e supporta DOM (Document Object Model). W3C, nella Recommendation del 4 Settembre 2001, scrive "SVG rappresenta un punto di incontro tra un linguaggio XML-based e la possibilità di creare grafica vettoriale bidimensionale". [*Testo Completo*]

Lo standard SVG è stato sviluppato con il supporto di tutte le maggiori società ed organizzazioni grafiche del Web: Adobe, Apple, AutoDesk, Bit-Flash, Corel, HP, IBM, ILOG, Inso, Kodak, Macromedia, Microsoft, Netscape, Oasis, Open Text, Oxford University, Quark, RAL, Sun-Microsystems, W3C e Xerox. Questo gli garantisce un vasto supporto e numerosi appoggi riguardo l'importazione/esportazione da/verso altri formati, e la distribuzione di visualizzatori e player.

Nel capitolo di introduzione del consorzio W3C, si legge :

> "*SVG is a language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images (raster) and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.*"

Che caratteristiche ha SVG?

Dall'Agosto al Settembre del 1999, ovvero in soli 2 mesi, W3C ha sviluppato ben 6 versioni diverse di SVG: nell'intera storia del consorzio non si era mai assistito alla produzione di un numero così elevato di release in un intervallo così limitato di tempo…

Dopo tutto, forse, non avevamo già abbastanza standard grafici per il WEB ? :

- GIF (Graphics Interchange Format)
- JPEG (Join Photographic Experts Group)
- PNG (Portable Network Graphics) anche se poco conosciuto
- WebCGM Profile diventato Recommendation W3C nel Gennaio 1999 ed utilizzato nel settore areospaziale
- senza poi dimenticare Macromedia Flash e Apple Quick Time supportati dalla maggior parte dei browser
- per terminare con DrawML, PGML (Precision Graphics Markup Language), HGML(Hyper Graphics Markup Language) e VML (Vector Markup Language).

Che farsene di un altro standard grafico?

Beh … anzitutto GIF, PNG e JPEG sono formati *bitmap* (ossia *raster*) che richiedono di memorizzare tutti i singoli pixel componenti un'immagine. Anche se questi formati utilizzano sofisticati algoritmi di compressione per ridurre le dimensioni dei file, per alcuni tipi di immagine, essi richiedono uno spazio significativamente superiore a quello della grafica vettoriale.

The real benefit of scalability is that entire pages will look the same on whatever screen they are viewed.

Sulle pagine W3C di Graphics Activity, Chris Lilley discute alcuni vantaggi di grafica scalabile a vettori, rispetto a una grafica bitmap. Le immagini bitmap sono dipendenti dalla dimensioni: appaiono diverse se visualizzate da diversi dispositivi di diversa capacità di risoluzione; l'immagine visualizzata a schermo e quella stampata possono differire l'una dall'altra. Lilley ha scritto

> *W3C is therefore developing a standard vector graphics format, SVG, written in XML and usable as an XML namespace, that matches the needs of content providers and browser vendors alike. It is designed to work well across platforms, output resolutions, color spaces, and a range of available bandwidths. Scalable Vector Graphics will mean that Web documents will be smaller, faster and more interactive, and be displayable on a wider range of device resolutions, from small mobile devices through office computer monitors to high resolution printers. This will be a significant advance in Web functionality*.

Nell'esempio riportato di seguito, la stessa immagine, codificata con due formati diversi, viene visualizzata prima nelle sue dimensioni originali, poi viene ingrandita di un fattore 3. Il formato bitmap (in questo PNG) non è affatto scalabile: la risoluzione dell'immagine rimane costante e, all'aumentare delle dimensioni di visualizzazione, l'immagine viene distorta.
Il formato vettoriale (in questo caso SVG) non usa una mappa di pixel, bensì una lista di comandi che disegnano le varie parti della figura; è questo che gli consente un'elevata scalabilità.

Small PNG image

Small SVG image

Enlarged PNG image

Enlarged SVG image

Ma allora che dire di Flash e QuickTime che invece sono formati a *grafica vettoriale*?
Non sono standard costruiti su XML e quindi non godono di tutti i benefici di tali linguaggi.

PGML, DrawML, HGML e VML? Anche se basati su XML, hanno altri problemi, in particolare non risultano completamente integrabili con tutti gli altri linguaggi della grande famiglia XML.

I due fattori principali che determineranno il successo o il fallimento di SVG sono

- La distribuzione di un elevato numero di player (su un numero sufficiente di piattaforme)
- La difficoltà insita nella creazione di un documento SVG.

Altra caratteristica di SVG è quella di essere costituito da semplice testo, che, come tale, può essere indicizzato, selezionato, ricercato ed estratto, conferendo massima flessibilità di accesso ai contenuti. Si, SVG è "**accessibile**", nel senso che in SVG tutto può essere collocato e identificato all'interno di un contesto.

Ciò può avere due interessanti applicazioni:

1. Riprendendo l'esempio della bicicletta, si può associare ad ogni componente (route, pedali, manubrio,…) una *descrizione testuale alternativa* (cfr. Descrizione Testuale Alternativa). In tal modo, la stessa informazione (bicicletta) può essere descritta utilizzando dispositivi non grafici, ottenendo così un'astrazione in grado di renderla accessibile a tutti i tipi di applicazione. L'informazione e **una** sua rappresentazione, vengono fuse nello stesso formato.

2. Il formato finale in cui visualizzare l'immagine può essere scelto dinamicamente in base al valore assunto da una serie di variabili. Se l'applicazione non consente il rendering delle immagini png, allora si utilizzerà il formato jpg, se no il gif, …

SVG è stilizzabile con l'uso dei CSS (Cascade Style Sheets); si tratta di fogli di stile a cui si può far riferimento per la formattazione degli elementi della pagina. A ciascuno di questi si attribuiscono delle caratteristiche (specificate nello stylesheet) di modo che, per modificarne l'aspetto, non si dovrà accedere al codice sorgente del documento, bensì basterà sostituire semplicemente il solo foglio di stile. Questa possibilità è utile sia al costruttore della pagina, che all'utente finale (che può adattare la visualizzazione del documento alle proprie esigenze).

SVG, consente la gestione di eventi, che, innescati da una qualsiasi azione e/o condizione, possono essere catturati e gestiti per eseguire le più svariate operazioni (la pressione del tasto del mouse sull'immagine può provocare, ad esempio, la comparsa di un campo di form).
Questo, unito all'utilizzo di DOM (Document Object Model), consente di poter utilizzare, ad esempio, JavaScript (linguaggio di scripting per altro già molto utilizzato nella realizzazione di pagine WEB) per accedere e manipolare a runtime tutti i componenti di una pagina SVG.

I documenti SVG possono, poi, essere inclusi in documenti scritti in altri linguaggi XML, e possono a loro volta includere frammenti scritti in altri linguaggi XML. Mixare diversi linguaggi di markup incrementa l'accessibilità del documento, che può essere quindi strutturato in varie parti, ciascuna descritta dal linguaggio più adatto. Ad esempio, un documento MathML può utilizzare SVG sia per visualizzare le equazioni descritte, sia per proporne il grafico.

In definitiva la chiave del futuro successo di SVG sta nei seguenti punti:

- File immagini (**mediamente**) più piccoli e tempi di download più brevi rispetto a quelli bitmap;
- Indipendenza dalla risoluzione e dal dispositivo utilizzato;
- Adattabilità a dispositivi con banda bassa e memoria limitata;
- Paning e Zooming (per visualizzare a diversi livelli di dettaglio);
- Completa compatibilità con XHTML: SVG può essere integrato all'interno di pagine WEB;
- Possibilità di catturare "eventi" e gestirli tramite qualunque linguaggio di scripting;
- Indicizzabilità del testo inserito nell'immagine, che quindi può essere oggetto di query;
- Possibilità di animazioni e trasformazioni complesse dell'immagine;
- Generazione dell'immagine anche a partire dai dati contenuti in un database (e non necessariamente memorizzati in modo statico su un file);
- Utilizzo di CSS (Cascading Style Sheets) per modificare la rappresentazione dell'immagine;
- Supporto dello standard RGB e, mediante l'uso di CSS, anche di una specifica di colori alternativa grazie all'utilizzo dei profili ICC (International Color Consortium);

Non meno importante: SVG è un Standard Aperto (Open Standard) . Gli standard *proprietari* tendono a favorire una specifica casa produttrice, creando così problemi di interoperabilità. Aggregandosi ad uno standard aperto, i produttori possono garantire l'intercambiabilità della grafica non solo fra applicazioni, ma anche fra piattaforme diverse. (cfr.Tipi di Software).

DOM

DOM è usato per modellare il contenuto di un documento o una pagina web, allo stesso modo in cui si modellerebbe un oggetto in un linguaggio di programmazione Object Oriented, consentendo di aggiungere, rimuovere e modificare gli elementi in esso contenuti.
Nonostante il nome possa trarre in inganno, DOM non è un modello per oggetti come ad esempio sono COM (Component Object Model) o CORBA (Comom Object Request Brokerage Architecture); DOM è costituito da un set di interfacce e non presenta alcuna implementazione concreta all'interno della sua Specification. Tali interfacce sono scritte in *Object Management Group Interface Definition Language* (OMG IDL), rendendo DOM indipendente dal linguaggio utilizzato per la programmazione.

Al momento esistono 3 diversi livelli di DOM:
Level 1: Incentrato sulla struttura di documenti HTML e XML. Consente accesso e manipolazione dei dati contenuti nei documenti.
Level 2: Questa seconda fase include il supporto dello style sheet (accesso e modifica dello style del documento). Definisce in oltre il modello ad eventi.
Level 3: Consente il caricamento ed il salvataggio dei documenti ed il supporto ai "modelli di contenuto" (ad esempio, Document Type Definition, DTD, in caso di documento XML) con relativa validazione.

Perché usare DOM?
Seguendo lo standard DOM, un'applicazione sarà in grado di manipolare i dati contenuti nel documento (es. scritto in XML) e, lo stesso codice, potrà essere utilizzato per altri componenti DOM. In oltre potendo scegliere di volta in volta una differente implementazioni di DOM sarà possibile utilizzare quella più adatta alle condizioni operative, ad esempio in caso di risorse di memoria limitate, sarà conveniente utilizzare un'implementazione di piccole dimensioni …
Altro vantaggio è che l'utente può usare il suo linguaggio preferito per accedere e manipolare le informazioni, e tale scelta non influenzerà affatto il risultato finale.

Un parser DOM XML è un'applicazione, ad esemio, JAVA, che converte un documento XML in qualche oggetto (Java). Dopo la fase di "parse" (analisi sintattica), una rappresentazione del documento stesso viene creata nella memoria della Java Virtual Machine (JVM), alla stregua di tutti gli altri oggetti utilizzati. Usando le specifiche dell'interfaccia DOM, possiamo accedere e manipolare i dati memorizzati in questi oggetti Java.

Si supponga di avere un documento XML come il seguente :

```
<?xml version="1.0"?>
      <book>
        <title>Design Patterns</title>
        <authors>
          <name>Erich Gamma</name>
          <name>Richard Helm</name>
          <name>Ralph Johnson</name>
          <name>John Vlissides</name>
        </author>
        <publisher>Addison-Wesley</publisher>
      </book>
      <book>
        <title>Managing the Software Process</title>
        <authors>
          <name>Watts S. Humphrey</name>
        </authors>
        <publisher>Addison-Wesley</publisher>
      </book>
</xml>
```

Utilizzando un (qualunque) parser, otteremo la seguente struttura dati :

```
[document]
 +--[node]book
 |    |
 |    +--[node]title="Design Patterns"
 |    +--[node]authors
 |    |    +--[node]name="Erich Gamma"
 |    |    +--[node]name="Richard Helm"
 |    |    +--[node]name="Ralph Johnson"
 |    |    +--[node]name="John Vlissides"
 |    |
 |    +--[node]publisher="Addison-Wesley"
 |
 +--[node]book
      |
      +--[node]title="Managing the Software Process"
      +--[node]authors
      |    +--[node]name="Watts S. Humphrey"
      |
      +--[node]publisher="Addison-Wesley"
```

E ora possiamo usare la NodeList (ad esempio il metodo "item" di NodeList) per accedere a ciascuno nodo ed ai suoi elementi. Modificando il contenuto di tali nodi, il documento si modificherà di conseguenza.
Sfortunatamente anche se nelle loro ultime versioni, sia Internet Explorer che Netscape, supportano Document Object Model, ci sono ancora parecchie incompatibilità fra i due browser, che utilizzano due diverse implementazioni di DOM; questo impone allo sviluppatore web, di utilizzare solo le API del DOM Level 1.

Com'è fatto SVG?

In questo capitolo vengono analizzate le caratteristiche del codice di un generico documento SVG.
SVG, essendo un linguaggio XML, utilizza un insieme di TAG per la descrizione dei propri elementi; alla pagina http://www.w3.org/TR/SVG è possibile trovarne una descrizione analitica e completa, mentre il Data Type Definition (che definisce le regole di produzione del linguaggio) è disponibile a: http://www.w3.org/TR/SVG/svgdtd.html.

In SVG tutto è un elemento grafico; e un elemento grafico può essere una generica shape, un testo o un riferimento ad un altro elemento grafico.

Graphics Elements = {Shapes , `text` , Graphics Referencing Elements}

- **Shapes** = Basic Shapes + `path`
  - **Basic Shapes**
    - `rect` (rettangolo, anche con angoli arrotondati)
    - `circle`
    - `ellipse`
    - `line`
    - `polyline`
    - `polygon`
  - `path`
- `text`
- **Graphics Referencing Elements**
  - `image`
  - `use`

Elemento	Attributi di Base	Descrizione
`Svg`	x, y, width4, height, allowZoomAndPan	Questo è l'elemento più esterno che contiene tutti gli altri elementi grafici e definisce un'immagine SVG. La posizione *(x,y)* risulta utile quando l'immagine è visualizzata all'interno di una pagine WEB. *width* e *height* rappresentano larghezza ed altezza, mentre il flag *allowZoomAndPan* consente all'utente di zoomare e ricentrare l'immagine (ossia: i rispettivi comandi del browser/plug-in risultano attivi).
`Rect`	x, y, width, height	Rettangolo di base *width* ed altezza *height*. E' possibile inoltre realizzare rettangoli con "spigoli arrotondati", specificando ulteriormente *rx*, *ry*, ovvero i raggi dell'ellisse usata per arrotondare gli angoli.
`Circle`	cx, cy, r	Circonferenza di centro *(cx,cy)* e raggio *r*.
`Ellipse`	cx, cy, rx, ry	Ellisse di centro *(cx,cy)* e raggi *rx, ry*.
`Line`	x1, y1, x2, y2	Linea retta descritta dai due estremi (punti iniziale e finale).
`Polyline`	Lista di punti	Lista di "coppie di coordinate *(x,y)* " Insieme di punti costituenti la curva, separati da una virgola, o da uno spazio. Esempio: "20,20 50,100 200,80 70,300"
`Polygon`	Lista di Punti Un polygon è una polyline, che viene automaticamente chiusa.	Lista di "coppie di coordinate *(x,y)* " Insieme di punti costituenti il poligono, separati da una virgola, o da uno spazio.
`Text`	String, x, y	Si riferisce specificamente ad una stringa che deve essere visualizzata, a differenza di quanto accade per gli elementi *desc e title* che hanno uno speciale significato e possono o meno essere visualizzati.
`Image`	x, y, width, height, xlink:href	Questo elemento consente di inserire un'immagine PNG o JPEG (ma non una GIF) all'interno di un'immagine SVG.

Un'altra particolarità è la possibilità di catturare eventi (12 diverse tipologie) a cui poi un Graphic Element può rispondere, come ad esempio `onclick`, `onmouseover`, `onkeydown`, e `onload`.

Quello che segue è un frammento di codice SVG:

```
<svg width="100%" height="100%">
    <g id="Oobj" transform="translate(290.00,229.50)">
        <text id="Gobj" class="dfltt" x="-97px" y="-21px"
             width="194px" height="42px">
             <tspan x="-97" y="-21" style="font-size:19px">
                  Welcome to my SVG web site.
             </tspan>
             <a xlink:href="iss62fig4.svg">
                  <tspan x="-97" y="6"
                        style="text-decoration:underline;fill:blue;">
                        Click here to see an animated SVG page
                  </tspan>
            </a>
        </text>
</svg>
```

<svg>
Tutto il codice SVG deve stare dentro al TAG <svg>, inoltre è possibile specificare due attributi width e height, che impostano, rispettivamente, in percentuale la larghezza e l'altezza dell'area visibile.

<g>
Il Tag <g> sottende molti altri tag tra cui <tspan> e un tag di link <XSLINK> Il Tag <g> è un elemento di raggruppamento che consente di assegnare proprietà comuni a diversi oggetti. In questo caso particolare tali oggetti condividono un identificatore "id" e una trasformazione “transform”. Di fatto <g> sortisce lo stesso effetto del tag di HTML, con la differenza che <g> consente anche di referenziare gli oggetti in un secondo tempo e cambiarne ad esempio la trasformazione.

Fill
Disegna la parte interna della figura a cui si riferisce. Si può specificare un colore, o richiedere che la figura risulti anche solo parzialmente trasparente.

Stroke
Disenga il contorno della figura. E' possibile controllare colore, spessore, antialiasing e opacità del contorno.

Il seguente esempio mostra l'utilizzo di stroke, e i meccanismi di ereditarietà e overriding degli attributi:

```
<svg height="50" width="100">
        <g style="stroke:black; stroke-width:2; fill:blue">
                <path d="M 10 10 L 30 10 L 30 40 L 10 40 Z"/>
                <path d="M 40 10 L 60 10 L 60 40 L 40 40 Z"/>
                <path style="fill:green" d="M 70 10 L 90 10 L 90 40 L 70 40 Z"/>
        </g>
</svg>
```

Tutti e tre i rettangoli verranno disegnati con un contorno nero, largo 2pixel, come specificato dall'attributo "style" dell'elemento <g>. I primi due rettangoli ereditano la proprietà "`fill:blue`" da <g>, mentre il terzo usa "`fill:green`", inserita nello stesso tag di <path> (override di style).

All'interno di questo gruppo di elementi troviamo anche il tag <text>. HTML assume che tutto ciò che si trovi nel documento e che non sia parte di un Tag, sia testo, e lo visualizza. SVG non fa questo genere di assunzione. Se si desidera visualizzare del testo, è necessario inserirlo esplicitamente all'interno di un tag <text>. Il Testo, in SVG, è considerato alla stregua di qualsiasi altro elemento grafico. Di conseguenza è possibile assegnargli un ID, uno Style e una Trasformation. E quindi, per fare un esempio, risulta immediato anche ruotare tale testo secondo un angolo in senso anti/orario.

<Text>, **<Tspan>**
Per visualizzare un testo in un'immagine SVG, si usta l'elemento <text>. Si specifica l'attributo style per definire le proprietà del testo, e si forniscono gli attributi x ed y per indicare al visualizzatore SVG dove andare a disegnare. All'interno di un elemento <text> è possibile aggiustare i valori delle coordinate x, y (sia specificando nuove coordinate assolute, che indicando la variazione rispetto alla posizione corrente) tramite l'uso del tag <tspan>.In SVG non esiste nulla del tipo fine-linea o fine-paragrafo. Se si vuole che il nuovo testo riparta dalla successiva riga, è necessario indicare manualmente tale spostamento, specificandone le coordinate. Questa scelta è dovuta al fatto che a differenza di HTML che fa uso intensivo di testo, SVG vuole trattare il testo come un oggetto grafico, e quindi by-passare tutte le operazione di formattazione utilizzate esplicitamente per esso. Un conto è fornire un supporto grafico (se pur minimo) a un sistema

Di seguito un frammento di codice che mostra l'utilizzo di <text>:

```
<svg height="50" width="100">
  <text style="font-size:36; font-family:Times Roman; fill:blue"
        x="10" y="40">
    The quick brown fox
  </text>
</svg>
```

L'attributo style dice al visualizzatore SVG di usare un font Times Roman, pica 36, e di riempire la parte interna delle lettere utilizzando il colore blu.
Questo è quanto si ottiene:

<a>

Questo tag definisce un anchor a un altro elemento SVG. La risorsa remota (destinazione del link) è definita per mezzo di un URI specificato dall'attributo href di XLINK. Tale destinazione può essere una qualunque risorsa del Web (un'immagine, un video clip, un programma, un altro documento SVG, un documento HTML, un elemento all'interno del documento corrente, …).

<animateMotion>

```
<animateMotion path="M 40 40 L 300 300" begin="1s" dur="6s"
fill="freeze"/>
```

Probabilmente la caratteristica più interessante degli oggetti grafici SVG è la possibilità di venire animati, tramite l'utilizzo del tag <animateMotion>.
In questo esempio è stato realizzata un'animazione molto semplice lungo un cammino che parte da x=40, y=40 e arriva a x=300,y=300. Gli elementi del gruppo iniziano a muoversi dopo 1 secondo, e terminano il percorso in 6 secondi. Al termine dell'animazione, gli elementi rimangono "congelati" (freeze) in modo tale da non poter tornare indietro.
In esempi più elaborati è possibile chiedere a tali elementi, anche di ruotare, percorrere distanze diverse, seguire cammini molto più complessi, ed altro ancora.

<path>

Questo elemento si utilizza per definire una forma. Un path contiene un certo numero di comandi che dicono al visualizzatore SVG di: traslare in un punto, disegnare una linea o una curva, chiudere una linea e riempirla di un colore. La maggior parte degli elementi SVG (<path>) compreso, prevedono un attributo style che utilizza proprietà CSS per definire le caratteristiche con cui gli oggetti dovranno essere formattati.

Di seguito un frammento di codice mostra l'utilizzo di <path>:

```
<svg height="50" width="100">
  <path style="stroke:black; stroke-width:2; fill:red"
        d="M 10 10 L 90 10 L 90 40 L 10 40 Z"/>
</svg>
```

L'attributo style specifica che il path deve essere realizzato con una linea nera di larghezza pari a 2 pixel, e riempito di colore rosso. Il significato dell'attributo `d` è mostrato nella seguente tabella.

SVG	Significato
M 10 10	Trasla il punto corrente a (10,10). M sta per `move`.
L 90 10	Disegna una linea dal punto corrente al punto (90, 10). L sta per `lineto`. Il comando `lineto` traccia la riga e muove di conseguenza il punto corrente.
L 90 40	Disegna una linea fino al punto (90,40).
L 10 40	Disegna una linea fino al punto (10, 40).
Z	Chiude il path e lo riempe con il colore specificato dal valore dell'attributo style.

Parametri dell'attributo <d>:

- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic bezier curve
- T = smooth quadratic bezier curveto
- A = elliptical arc
- Z = closepath

Il risultato che si ottiene è :

Vediamo infine come tracciare curve e archi, utili ad esempio per rappresentare diagrammi a torta. Questa operazione risulta già più complessa perché si richiedono funzioni trigonometriche (i classici 'seno' e 'coseno') che non sono disponibili né in XSLT né in Xpath. Ciò ci costringe a creare nuove funzioni che vanno ad aumentare il potere espressivo del processore XSLT.

```
<svg height="200" width="200">
  <path style="fill:blue; stroke:black; stroke-width:2;
               fillrule:evenodd; stroke-linejoin:miter;"
        transform="translate(100, 100)
                   rotate(-72.24046757584189)"
        d="M 80 0 A 80 80 0 0 0 25.26622959787925
           -75.90532024770893 L 0 0 Z"/>
</svg>
```

In questo esempio, usiamo l'attributo `transform` per muovere (traslare) l'origine (il punto 0,0) fino a 100,100 e per ruotare l'intero sistema di riferimento di 72 gradi in senso orario. Possiamo usare la stessa tecnica per disegnare le varie fette del grafico a torta, riuscendo così facilmente ad incentrarle tutte nel punto 0,0 e semplificando notevolmente il "lavoro trigonometrico". Vediamo brevemente gli argomenti dell'attributo `d`

SVG Data	Meaning
M 80 0	Muove il punto corrente a (80, 0).
A 80 80 0 0 0 25.266... -75.905...	Traccia un arco ellittico dal punto corrente fino ad un determinato altro punto. I primi due numeri (80,80) rappresentano i due raggi dell'arco (se sono uguali, l'ellisse degenera in una circonferenza). I successivi tre numeri (0,0,0) influenzano la rotazione, e non li prenderemo in considerazione. Gli ultimi due numeri (25.266... -75.905...) sono le coordinate del punto finale dell'arco.
L 0 0	Traccia una linea dal punto corrente (la fine dell'arco che abbiamo appena disegnato) fino all'origine.
Z	Chiude il cammino e lo riempe con il colore settato a suo tempo `style`.

Quello che si ottiene è :

Gli elementi grafici di un documento SVG appaiono (vengono disegnati) nello stesso ordine in cui si presentano nella descrizione. Se traccio un quadrato, e nella riga successiva chiedo di disegnare nello stesso punto un cerchio, allora il quadrato apparirà in background, mentre il cerchio verrà disegnato in foreground.

<title> e **<desc>**

Il rendering grafico di un oggetto SVG, in alcune circostanze, può risultare complicato, se non impossibile, o addirittura inappropriato. Si pensi ai piccoli dispositivi palmari con risorse grafiche limitate, o utenti con difficoltà visive. Per questo motivo SVG fornisce la possibilità di gestire rappresentazioni alternative delle informazioni. Questo, solitamente, si traduce con la presenza di ulteriore testo, allegato ai vari elementi componenti l'immagine, che si presta ad essere visualizzato o ulteriormente processato.

*L' **accessibilità** di un documento permette di navigare fra le sue caratteristiche, ed estrarne le informazioni ritenute necessarie. Gli oggetti che ci circondano nella realtà di tutti i giorni posseggono enormi quantità di tali informazioni, che noi siamo abituati a acquisire tramite i nostri cinque sensi di percezione.*

Così, nella nostra mente, una rosa possiede un colore, una forma, un odore, una consistenza, una sensazione al tatto, ... Ovviamente non utilizzeremo sempre tutti questi dati (che la nostra memoria ci fornisce) per descrivere l'oggetto 'rosa'; dipendentemente dal livello di dettaglio, o dal tipo di conversazione che stiamo sostenendo, sceglieremo di elencare solo i particolari che ci sembrano più opportuni.

Ciò che facciamo, quindi è selezionare dal quadro generale, solo alcuni aspetti.

Allo stesso modo, allegando più informazioni possibili al formato di descrizione di un oggetto, saremo in grado di produrne diverse rappresentazioni, una per ogni contesto nel quale ci muoviamo.
***Più informazioni vengono codificate, maggiore è il numero di richieste a cui si può far fronte**.*

Il modo più semplice di specificare un testo ("alternative text"), è quello di includere i seguenti tag in ogni sottoelemento:

`title`

Fornisce un titolo per l'elemento che lo contiene. La stringa di testo può essere visualizzata graficamente, così come processata da un sintetizzatore vocale, o da un dispositivo per la conversione in codice Braille, ….

`desc`

Fornisce una descrizione più dettagliata e completa dell'elemento che la contiene.

Esempio:

```
<svg>
     <g id="ComputerA">
          <title>Computer A</title>
          <desc>A common desktop PC</desc>
          <path d="… … …" />      < !-- Drawing -- >
      </g>
      <g id="ComputerB">
          <title>Computer B</title>
          <desc>A common desktop PC</desc>
          <path d="… … …" />      < !-- Drawing -- >
     </g>
     <g id="CableA">
          <title>Cable A</title>
          <desc>10BaseT twisted pair cable</desc>
          <path d="… … …" />      < !-- Drawing -- >
     </g>
</svg>
```

<tref> Reusing Alternative Text as Graphics

Nei formati raster, il testo è convertito in pixel; SVG usa il tag `text`, e lo mantiene nella sua forma originaria. Inoltre è possibile riutilizzare il testo "alternativo" dei tag title e desc, tramite semplice riferimenti

```
<svg>
     <g>
        <title id="hub">Hub</title>
        <desc>A typical 10BaseT/100BaseTX network hub</desc>
        . . .      . . .     . . .
        <text x="0" y="-10">
              <tref xlink:href="#hub"/>
        </text>
     </g>
</svg>
```

<use> Reusing Graphic Component

SVG utilizza costrutti per il riutilizzo di componenti grafici. Questo facilita la comprensioni di strutture di immagini anche complesse, in quanto tali componenti riutilizzabili vengono descritti per intero (e quindi devono essere compresi dall'utente che ne legge il codice) una, ed una solta volta, e linkati con opportuni riferimenti laddove ce ne sia bisogno. Questo aspetto diventa di particolare importanza quando l'immagine non viene visualizzata, ma si utilizzano dei meccanismi di rappresentazioni alternativi (ad esempio un output vocale o braille, in cui la ripetizioni di un sottoinsieme di informazioni, comporta, all'end user, tempi di acquisizione maggiori).

```
<svg>
      <defs>
            <symbol id="hubPlug">
                  <desc>A 10BaseT/100baseTX socket</desc>
                  <path d="M0,5 h5 v-9 h12 v9 h5 v16 h-22 z"/>
            </symbol>
      </defs>
      <g>
            <title>Socket</title>
            <use xlink:href="#hubPlug"/>
    </g>
</svg>
```

Oltre a <title> *e* <desc>*, altro meccanismo per migliorare l'accessibilità di un'immagine SVG è quello di descriverne il contenuto mediante metadati. I metadati sono "dati che descrivono dati". Si veda la descrizione di RDF (cfr.RDF) e la sua integrazione con SVG.*

Come si usa SVG?

SVG è "*text based*", ovvero è composto da semplice testo, e come tale, un file SVG può essere realizzato utilizzando un comunissimo editor di testo. Questo rappresenta da un lato un vantaggio, e dall'altro una delle sue (poche) limitazioni; infatti, esattamente come accade anche per HTML, ogni sviluppatore che conosce le regole del formato, può creare, e modificare il testo SVG.

Non appena ci si addentra in un codice SVG ci si accorge subito delle incredibili potenzialità di questo formato, in grado di fornire allo sviluppatore moltissime possibilità che neppure linguaggi come HTML o DHMTL offrivano.
Per altro, DHTML ha dimostrato di essere **troppo complesso** per essere creato direttamente (ovvero senza l'ausilio di tool dedicati), per la per la stragrande maggioranza dei suoi utilizzatori (tipicamente i web developers); SVG, da questo punto di vista, risulta ancora più vasto e complicato di DHTML !!! Sorge quindi la domanda : "cosa ci si aspetta che uno sviluppatore costruisca con SVG che non sia già in grado di realizzare con HTML?".

Ciò di cui mi sto convincendo è che proprio rispondendo a questo interrogativo saremo veramente in grado di prevedere la diffusione e la concreta possibilità di SVG di affermarsi come standard. E questo non perché SVG non presenti nuove caratteristiche rispetto a HTML, bensì perché l'accesso a tali contesti va decisamente oltre le reali capacità della maggior parte degli sviluppatori.
(SVG è troppo difficile per essere vero ?!?!?)

Cercando un riscontro sull'argomento, ho raccolto, in un forum di discussione, diversi pareri; in particolare quello di un autore, da anni sviluppatore di animazione grafica per il Web, che esprime con franchezza un parere sullo scontro che vede scendere in campo da un lato i software e gli standard ormai consolidati ed utilizzati quotidianamente dalla maggior parte degli progettisti, e dall'altro, le nuove proposte, che, in un evidente tentativo di soppiantare le vecchie tecnologie in nome di una standardizzazione del Web, mettono in discussione l'intera struttura su cui per diverso tempo, si è fatto affidamento. Quello che segue è un estratto che riassume in poche righe l'intera discussione:

" I personally would pick SVG because it has been carefully developed as a standardized open format that works in conjunction with many other standard formats such as XML, SMIL and HTML. But as an Internet graphics animation developer, my opinion is probably biased. I obviously would prefer access to a format I can work with and use."

Un'interessante alternativa per creare codice SVG, deriva dalla possibilità di generare immagini in tempo reale, partendo dai dati contenuti in un database sorgente, con l'ausilio di un linguaggio di trasformazione.
Ad esempio si supponga che un web server sia collegato a una base di dati contenente le vendite giornaliere di una serie di società, e quindi aggiornati frequentemente. Usando XML come linguaggio di descrizione di tali dati, una semplice trasformazione XSLT genererà, **on the fly**, splendidi grafici e diagrammi, rappresentanti i vari aspetti delle vendite.

Si immagini come, allo stesso modo, l'utente potrà scegliere progressivamente il livello di dettaglio per consultare un progetto meccanico, o un referto medico, sempre sfruttando lo stesso file dati di partenza, ma generando, di volta in volta, solamente delle "viste" diverse.

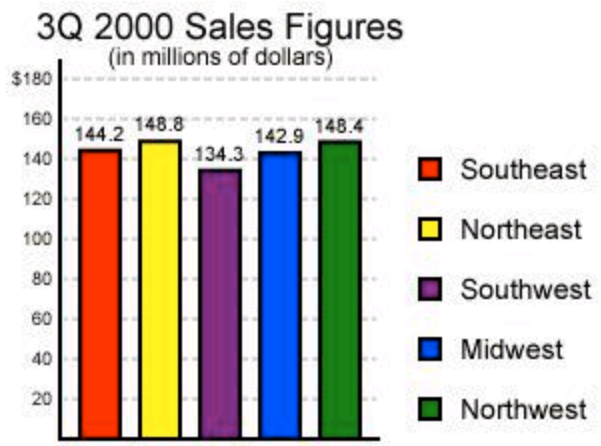
Un esempio pratico potrà chiarire questo discorso …

Si prenda un file XML contenente dei dati strutturati in qualche maniera; in questo esempio utilizziamo proprio i dati di vendita di alcuni prodotti, divisi per regione di vendita.

```
<sales>
  <caption>
    <heading>3Q 2000 Sales Figures</heading>
    <subheading>In millions, broken down by region</subheading>
  </caption>
  <region>
    <name>Southeast</name>
    <product name="Heron">38.3</product>
    <product name="Kingfisher">12.7</product>
    <product name="Pelican">6.1</product>
    <product name="Sandpiper">29.9</product>
    <product name="Crane">57.2</product>
  </region>
  <region>
    <name>Northeast</name>
    <product name="Heron">49.7</product>
    <product name="Kingfisher">2.8</product>
    <product name="Pelican">4.8</product>
    <product name="Sandpiper">31.5</product>
    <product name="Crane">60.0</product>
  </region>
      … … … … … … …
</sales>
```

Si costruisca ora il codice XSLT che trasformi tali dati, producendone una rappresentazione. Tra le infinite rappresentazioni che si possono scegliere (e i numerosi formati in cui le stesse si possono codificare) scegliamo SVG.
Tramite l'utilizzo di template, che matchano con i tag dei dati XML, XSLT produce i vari comandi di visualizzazione SVG, che poi, terminata la trasformazione, potranno essere visualizzati con l'apposito player.

Questo è **un** possibile risultato finale:



Di seguito è riportato un breve estratto del codice XSL relativo alla trasformazione:

Root Element Template
Specifica chi deve creare la radice dell'albero finale. In questo caso dice al processore XSL di partire con l'elemento "sales".

```
<xsl:template match="/">
  <xsl:apply-templates select="sales"/>
</xsl:template>
```

<sales> template
Crea l'intero grafico SVG. Il titolo viene preso dall'elemento <heading> e dimensioni e posizioni delle barre del grafico sono calcolate sommando i fatturati dei vari prodotti, all'interno di ciascuna regione.

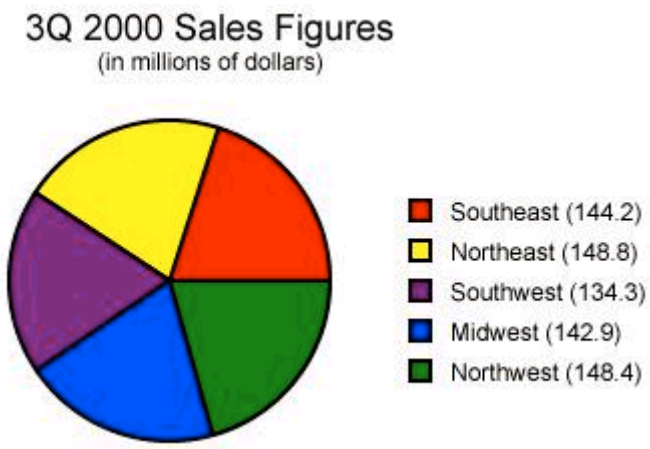
```
<xsl:template match="sales">
  <svg width="400" height="300">
    <text style="font-size:18; text-anchor:middle" x="120" y="20">
      <xsl:value-of select="caption/heading"/>
    </text>

<text style="font-size:12; text-anchor:middle"
        y="33" x="120">(in millions of dollars)</text>

      <g style="stroke-width:2; stroke:black">
        <path d="M 40 220 L  40  30 L 40 220 L 200 220 Z"/>
      </g>
```

Ora, per costruire le varie barrette, abbiamo bisogno dei seguenti parametri :
1) Le coordinate x,y a cui la barra inizia
2) Il colore con cui riempirla
3) La coordinata y a cui inserire l'etichetta della somma del venduto
Useremo le variabili $x-offset, $y-offset, $color, e $y-legend-offset, e le aggiorneremo ad ogni iterazione (il ciclo viene ripetuto per ogni istanza del tag <region>)

```
      <xsl:for-each select="region">

        <xsl:apply-templates select=".">

          <xsl:with-param name="color">
            <xsl:choose>
              <xsl:when test="(position() mod 5) = 1"> <xsl:text>red</xsl:text> </xsl:when>
              <xsl:when test="(position() mod 5) = 2"> <xsl:text>yellow</xsl:text> </xsl:when>
              <xsl:when test="(position() mod 5) = 3"> <xsl:text>purple</xsl:text> </xsl:when>
              <xsl:when test="(position() mod 5) = 4"> <xsl:text>blue</xsl:text> </xsl:when>
              <xsl:otherwise> <xsl:text>green</xsl:text> </xsl:otherwise>
            </xsl:choose>
          </xsl:with-param>

          <xsl:with-param name="y-legend-offset">
            <xsl:value-of select="60 + (position() * 30)"/>
          </xsl:with-param>

          <xsl:with-param name="x-offset">
            <xsl:value-of select="30 + (position() * 30)"/>
          </xsl:with-param>

          <xsl:with-param name="y-offset" select="220"/>
        </xsl:apply-templates>

      </xsl:for-each>
    </svg>
  </xsl:template>
```

Il <region> template (omesso), infine disegnerà la barra per ogni regione, con quell'altezza e di quel colore.

Se poi, al posto dell'istogramma, volessimo ottenere un grafico a torta, basterebbe cambiare il modulo XSLT che effettua la trasformazione (lasciando invariato tutto il resto), per ottenere questa nuova rappresentazione :



SVG e JAVA !

I precedenti due diagrammi sono stati ottenuti mediante due diverse trasformazioni XSLT su una stessa base di dati (descritta ovviamente in XML). Si sono applicati appositi tamplate in modo da disegnare per ogni "regione di vendita" una colonnina o un settore circolare di area proporzionale alla somma delle vendite della regione stessa. Il secondo grafico, però, nasconde qualche piccola difficoltà in più rispetto al primo, visto che né XSLT né Xpath prevedono l'uso di funzioni trigonometriche …
E' stato quindi necessario affidarsi a "funzioni esterne" ovvero a **extension functions**.

Per ottenere infatti le funzioni di seno e coseno, dobbiamo ricorrere a funzioni di estensione ed accedervi dal nostro stylesheet. Fortunatamente la classe `java.lang.Math` fornisce dei metodi statici per il calcolo di seno e coseno; inoltre **XALAN**[1] consente di accedere molto facilmente ai metodi definiti in una qualsiasi classe java. Per rendere tutto questo possibile, aggiungiamo alcune dichiarazioni `<xsl:stylesheet>`:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:javaMath="http://xml.apache.org/xslt/java"
  xmlns:lxslt="http://xml.apache.org/xslt"
  extension-element-prefixes="javaMath"
  exclude-result-prefixes="lxslt">
```

Questo frammento di codice definisce una coppia di namespaces, `javaMath` and `lxslt`, ed usa gli attributi standard XSLT `extension-element-prefixes` e `exclude-result-prefixes` per indicare al preprocessore XSLT che ogni elemento con questi prefissi non dovrà essere inviato all'output stream.

Il prossimo passo è quello di introdurre un elemento specifico Xalan che definisca le funzioni vogliamo utilizzare nel nostro stylesheet.

```
<lxslt:component prefix="javaMath" functions="cos sin toRadians">
```

Questo dice a Xalan che il prefisso del namespace `javaMath` deve essere associato alla classe Java `java.lang.Math`. Questa definisce le tre funzioni, `cos`, `sin`, and `toRadians`. Usiamo anche il metodo `toRadians` perché entrambe le funzioni `cos`, `sin` richiedono di avere come argomenti dei radianti.

Quindi la funzione di nome `javaMath:cos()` dice a Xalan di invocare il metodo `cos` della classe `java.lang.Math`. Un ultimo piccolo dettaglio: siccome l'attributo `src` dell'elemento `<lxslt:script>` inizia con "`class:`", Xalan capisce che i suddetti metodi sono statici, e quindi che non dovrà creare nessuna istanza di alcun oggetto per invocare tali funzioni.

Ora che abbiamo definito tutto, possiamo tranquillamente richiamare le funzioni di estensione:

```
<xsl:attribute name="d">
  <xsl:text>M 80 0 A 80 80 0 0 0 </xsl:text>
  <xsl:value-of select="javaMath:cos($currentAngle) * 80"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="javaMath:sin($currentAngle) * -80"/>
  <xsl:text> L 0 0 L 80 0 </xsl:text>
</xsl:attribute>
```

Tutto ciò può sembrare forse un po' laborioso, ma in questo modo siamo in grado di aumentare la capacità espressiva del processore XSLT senza dovere scrivere una sola riga di codice Java. Semplicemente conoscendo il nome dei metodi che ci interessano, con questo meccanismo possiamo utilizzarli in maniera del tutto trasparente.

[1] Apache ha sviluppato un package di nome XALAN che realizza un'interfaccia bidirezionale tra l'ambiente di programmazione JAVA e il formato XML. Le API di XALAN sono disponibili a xml.apache.org/xalan/index.html.

Quanto occupa SVG?

Le immagini SVG sono veramente più compatte e meno voluminose delle corrispondenti GIF,JPEG, e PNG?

Riportiamo questo piccolo test realizzato usando diversi esempi di immagini (riportate a fine pagina), e confrontando le dimensioni (in KByte) dei rispettivi file

File	Immagine	SVG	GIF	JPEG	PNG
Shapes.svg	shapes.gif	2	26	24	21
gradients.svg	gradients.gif	2	62	35	41
tiger.svg	tiger.gif	96	52	56	111

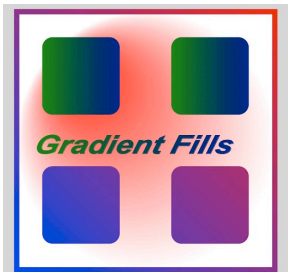
Le immagini SVG sono state visualizzate con CSIRO SVG Viewer e catturate e convertite in GIF, JPEG, e PNG usando Paint Shop Pro.

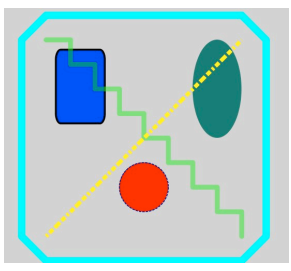
Anche se basato su un ristretto training set di oggetti, questo risultato consente di affermare che per alcuni tipi di immagini (le prime due), lo standard SVG consente un notevole risparmio di memoria (file 10-30 volte più piccoli dei corrispondenti bitmap). Per altre, invece, (tiger.svg richiede 700 linee di testo, che descrivono pattern anche molto complessi), SVG raggiunge dimensioni quasi doppie rispetto a Standard tradizionali.

Ciò nonostante, essendo SVG un formato testuale (ed il testo facilmente comprimibile) è possibile zipparlo, ottenendo elevati rapporti di compressione; in questo modo, anche le dimensioni dell'ultimo file, scendono al di sotto della media dei formati bitmap (raggiungendo, circa, 32KB).

Per approfondimenti sulla minimizzazioni delle dimensioni dei file SVG, si veda Minimizing SVG File Sizes.

Gradients *Shapes* *Tiger*





Bitmapped graphics vs. Vector graphics …

"*… per un browser è molto più semplice tracciare una circonferenza di raggio 100 pixel e centro nel punto 200,200 che non disegnare tutti i singoli punti, descritti ad uno ad uno, che la compongono (come si farebbe in una bitmap) …*"

I formati bitmap trattano ogni immagine come una collezione di punti, ed assegnano un colore ad ogni specifico pixel.

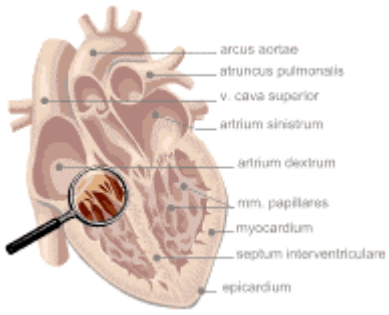
Tale tecnica , che per altro dà ottimi risultati quando l'immagine è visualizzata alla dimensione e con la risoluzione originale, è del tutto inadeguata per ingrandimenti e zoom.

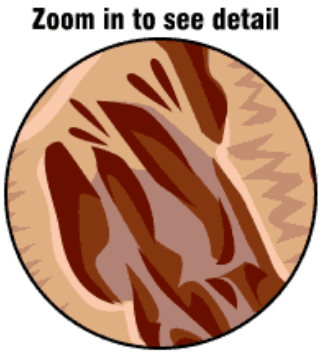
Bitmap

Vector

La grafica vettoriale, definisce l'immagine attraverso l'uso di figure geometriche (shape) e linee curve (path). Questo significa che, ad esempio, l'immagine di una retta obliqua potrà essere ingrandita un numero arbitrario di volte, senza presentare l'effetto di "gradinatura" tipico delle bitmap.

Le bitmap sono quindi adatte per grafica statica (ossia con limitata modificabilità). La grafica vettoriale, invece, consente di visualizzare l'immagine ad un qualsiasi livello di dettaglio, sempre utilizzando lo stesso file sorgente.





Zoomando consecutivamente nel punto di interesse si otterrà di volta in volta un disegno sempre più particolareggiato. Ogni singolo componente può poi essere modificato, ri-colorato o rimosso, a seconda delle esigenza

Il termine "vettore" è stato acquisito dall'Analisi Vettoriale, e risale ai tempi in cui i plotter producevano grafici semplicemente definendo alcune coordinate, e tracciando rette fra tali punti, per poi riempire di colore le figure così disegnate. La grafica vettoriale consiste in una serie di comandi che descrivono le istruzioni per comporre l'immagine desiderata.

La maggior parte dei dispositivi, inclusi monitor, stampanti a matrici di punti e laser, sono dispositivi raster (con l'ovvia eccezione dei plotter). Ciò significa che tutti gli oggetti, compresi i vettori, devono subire una sorta di traduzione in formato bitmap, prima di poter essere inviati a tali dispositivi. La differenza fra la grafica vettoriale e quella raster, da questo punto di vista, è che la grafica vettoriale viene tradotta in bitmap solo quando diventa realmente necessario (il più tardi possibile). Dopo tale operazione, dimensioni e risoluzione dell'immagine vengono fissati per sempre.

Come già discusso in precedenza, la grafica vettoriale, per propria natura, risulta più "compatta" da un punto di vista di occupazione di spazio. Ad esempio si tenti questo esperimento:

(a) usando un qualsiasi programma di disegno si tracci una linea diagonale dall'angolo in basso a sinistra, fino a quello in alto a destra, di una finestra di 640x480.
(b) Salvando l'immagine usando il migliore coefficiente di compressione possibile, le dimensioni del file risultante saranno prossime ai 3 Kbyte.
(c) La stessa linea, in grafica vettoriale, richiede un minimo di 4 Byte per il primo punto, 4 per il secondo estremo, e 3 per il valore RGB.
(d) Anche aumentando la precisione, il vettore potrà essere contenuto in non più di 24 Bytes.

D'altro canto, però:
- il vettore, a differenza del bitmap, richiede un elaborazione da parte del processore prima di poter essere visualizzato.
- Inoltre in bitmap si possono visualizzare immagini molto complesse, estremamente difficile (se non addirittura impossibili) da realizzare utilizzando vettori. Le illustrazioni vettoriali richiedono un'elaborazione notevolmente superiore rispetto alle bitmap. La visualizzazione di un volto umano può richiedere anche un paio di ore di elaborazione, mentre una bitmap può essere scannerizzata, salvata e visualizzata nel giro di qualche minuto.

Su questi nuovi criteri, allora, bisogna anche aggiungere che:
- anzitutto un vettore può essere visualizzato prima ancora che il corrispondente bitmap abbia ancora finito di essere scaricata dalla rete (**progressive rendering**),
- inoltre i vettori sono utilizzati per costruire immagini di complessità elevata; basti pensare che come il film "Jursassic Park Dinosaurs", di S.Spielberg così anche tutti gli effetti speciali 3D, sono interamente ralizzati in grafica vettoriale.

Un'altra caratteristica delle immagini in grafica vettoriale, è quella di poter essere modificate con estrema facilità. Nell'esempio sopra, una coordinata può essere sostituita con un'altra semplicemente modificando 4 byte. Con qualche byte in più per identificare l'operazione di traslazione, il vettore può essere completamente ridisegnato. Questa caratteristica è fondamentale se si vuole realizzare un'animazione.

Le immagini Bitmap sono ideali per rappresentare foto (real life images), o per essere utilizzate in software per il disegno; i vettori sono quanto di meglio si possa usare, per tutti gli altri scopi.

Il punto di forza della grafica vettoriale ("vector based" graphic), è sicuramente la sua alta scalabilità; tale caratteristica consente al singolo file sorgente di essere ri-dimensionato e visualizzato in qualsiasi scala. Di fatto, questo risulta un requisito fondamentale per icone di dimensioni variabili, come ad esempio i loghi di società ed aziende che possono essere riutilizzati in qualsiasi contesto, dal titolo sulla pagina Web, al marchio stampato su carta intestata.

Avversari ed alleati di SVG

Prima dell'arrivo di SVG, Adobe e Microsoft avevano già avanzato alcune proposte (rispettivamente PGML e VML) indirizzate all'utilizzo di grafica vettoriale.

- VML non è mai diventato una Recommendation W3C nonostante il grosso impegno di Microsoft; attualmente, l'unico browser in grado di visualizzare grafica VML è Internet Explorer 5.
- PGML, da parte sua, è stato sviluppato da Adobe e sostenuto da IBM e Sun.

W3C ha analizzato attentamente sia PGML che VML e ha preso la decisione più logica, ovvero ha estratto il meglio da entrambi i formati ed ha realizzato un nuovo standard, chiamato appunto SVG, supportato sia da PGML che da VML.

Per ulteriori approfondimento in merito alle similitudini esistenti fra VML e PGML, è possibile consultare il seguente articolo : VML/PGML Head-to-Head Comparison..

Ci verrebbe da chiederci, per quale motivo abbandonare formati come **Flash** e **QuickTime**, per abbracciare una nuova tecnologia come SVG. Bene, nonostante sia vero che SVG riproponga molte caratteristiche già presenti in entrambe, bisogna anche ammettere che offre altri vantaggi concreti. Essendo SVG un formato open-source, può essere semplicemente integrato con CSS, JavaScript, CGI e altre tecnologie universali di uso su Web. In oltre SVG è basato su Unicode, proprio come il testo presente in un comune documento HTML, e ciò lo rende di facile lettura ed indicizzazione da parte dei motori di ricerca di Internet.

> *"... Flash relies on proprietary technology that is not open source, at least not in the sense that we can right click on the page and see what is happening behind the scenes. SVG by contrast is open source and developers can readily learn from other developer's efforts in this area... "* (The Art is in the Code , *Eddie Traversa*)

Ma il grande vantaggio di SVG è rappresentato dal suo essere conforme alle XML Namespace Recomendation. Questo, assieme al supporto di DOM, dà allo sviluppatore la possibilità di controllare e maneggiare qualsiasi oggetto taggato a cui sia stato associato un attributo di nome, e quindi alla grafica e ai dati, di riflettere - e reagire- run time agli eventi innescati dall'utente.

SVG potrebbe risultare altamente competitivo anche nei confronti di un prodotto come Flash, già profondamente radicato nel settore. Una diffusione di tool SVG e un maggiore supporto da parte dei browser potrebbero esserne la chiave del sorpasso; ma fino a quel momento difficilmente si riuscirà a distogliere l'attenzione da un software potente e versatile come Flash, per dare spazio allo standard open-source del consorzio W3C.

Visualizzatori SVG

Adobe distribuisce un beta plug-in (ultima versione: 8.01) disponibile sul sito. Questo plug-in, disponibile sia per utenti Mac che PC, permette di visualizzare immagini SVG all'interno del browser.

IBM offre viewer app disponibile sul sito Alphaworks.

Corel ha realizzato un public beta plug-in per gli utilizzatori di CorelDRAW 9.0.

CSIRO, una società australiana operante nel settore delle scienze matematiche e dell'informatica, offre gratuitamente un Java-based viewer, unitamente ad un toolkit open-source.

Software per SVG

Apache® Batik™ **Ultima release : Batik 1.5 beta 1, 11 Marzo 2002**.
Batik è un tookit basato su tecnologia Java per applicazioni che utilizzano immagini in formato Scalable Vector Graphics (SVG) per tutti gli scopi: dalla visualizzazione, alla generazione, alla manipolazione.
Il progetto è composto da una set di moduli che possono essere usati singolarmente o congiuntamente a seconda delle specifiche soluzioni. Esempi di tali moduli sono : *SVG parsers*, *SVG generator e SVG DOM implementation*.
Con Batik è possibile maneggiare documenti SVG ovunque sia disponibile Java, poi, utilizzando ad esempio *SVG generator*, un'applicazione Java può esportare le informazioni prodotte, in formato SVG. Un'altra possibilità offerta da questo pacchetto, è quella di convertire SVG in vari formati, tra cui ad esempio JPEG e PNG.

Jasc® WebDraw™ **The Complete Solution for Creating SVG Graphics and Animation**
Un'unica soluzione per la progettazione di grafica e la generazione di codice sorgente del nuovo e potente fomato grafico del Web. Veloce, flessibile ed affidabile, **Jasc® WebDraw™** offre la possibilità di:

- Utilizzare tool di sviluppo per creare grafica ed animazioni di alta qualità
- Aggiungere sofisticati effetti di gradient, pattern e filtri per arricchire gli oggetti
- Compilare direttamente il codice SVG, per ottenere un miglior controllo e una ma flessibilità dei file creati
- Automatizzare la validazione del codice prodotto
- Visualizzare immediatamente ogni cambiamento apportato al sorgente
- Importare, modificare ed ottimizzare file SVG creati da altri programmi

SVG MapGen **A standalone tool for converting ESRI Shape and MapInfo MID/MIF documents to interactive SVG maps**
ESRI (Environmental Systems Research Institute) società californiana leader nello sviluppo di software GIS.
MID (Metafile for Interactive Documents), MIF (MapInfo Interchange File): formati per la rappresentazione di mappe.
Realizzato in C++, è un veloce e robusto convertitore di mappe in formato SVG.

Dove si usa SVG?

" … A language is only as good as the applications that can be written in it. … "
(P. Kanthman, XMLization of Graphics)

- GRAFICA PER IL WEB.

La realizzazione di grafica per il Web, è forse, uno degli aspetti più rilevanti che ha portato alla nascita di SVG. I formati attualmente già in circolazione presentano alcune limitazioni che il consorzio W3C ha tenuto in seria considerazione durante la progettazione del nuovo standard:

- **Carenza di Open Standard**
  C'è un discreto numero di formati di grafica vettoriale per il Web, così detti di proprietà (cfr. Tipi_Software) quali, ad esempio Macromedia Flash e Apple QuickTime. L'assenza di standard per la grafica vettoriale per il web ha, in parte contribuito, alla nascita di SVG.

- **Carenza di Interoperabilità**
  I comuni formati di grafica vettoriale non consentono interazione con gli altri standard orientati al Web (e spesso neppure con il documento, tipicamente HTML, in cui sono contenuti) e viceversa.

- **Carenza di interazione macchina-macchina**
  I formati di grafica vettoriale esistenti, come ad esempio, Macromedia Flash e Apple QuickTime, mirano a realizzare un meccanismo di comunicazione fra uomo e macchina. Questo, di certo, è requisito estetico primario nella progettazione pagine Web, ma assume implicitamente che il destinatario delle informazioni presentate sia un operatore umano, escludendo automaticamente la possibilità che tali dati siano invece destinati ad un'altra macchina (es: scambio di immagini fra due database).
  Inoltre non c'è alcun concetto di *validazione* automatica per immagini create con tali formati, né di conformità, ovvero di sicurezza che un'immagine creata con un tool funzioni anche con un altro. Per questo, i formati attuali ostacolano la comunicazione e l'interoperabilità macchina-macchina.

Oltre ad ovviare a queste problemi, SVG, nello scenario della grafica per il Web, propone anche nuove caratteristiche:

- **Flessibilità**. Diversamente da quanto accade in HTML, XML tiene completamente separati i **dati** dalla loro **rappresentazione**. Descrivendo le informazioni in termini di dati strutturati, un'applicazione utente può utilizzare la stessa pagina XML sia visualizzandone in qualche modo il contenuto (alla stregua di HTML), sia utilizzando i dati presenti per una generica operazione. Ad esempio: un dato taggato come numero di telefono può essere sia inserito nel logo della ditta a cui si riferisce, sia composto su di una tastiera di un telefono collegato, a seconda dei dispositivi e delle applicazioni che processano il file stesso.
  Queste potenzialità fanno inabissare HTML rispetto ad SVG.

- **Estetica.** Come già evidenziato più volte, la grafica "raster" mostra diversi punti deboli soprattutto quando cambiano le condizioni di visualizzazioni delle immagini (tipo di dispositivo, dimensioni, risoluzione, …), cosa che non accade, ovviamente, per la grafica vettoriale.

- **Elevato Contenuto.** SVG è scritto come una qualunque applicazione XML, ed utilizza il namespace XML. Questo gli consente di poter essere utilizzato assieme ad una qualsiasi altra applicazione XML, come ad esempio XHTML, SMIL e MathML. Ciò attribuisce a SVG la possibilità di realizzare layout di pagine estremamente sofisticati, e ricchi di elementi eterogenei fra loro, dando vita a pagine web molto complesse senza, per altro, aumentare la dimensione dei file che le memorizzano.

- **Uso di Stili.** I vantaggi dell'utilizzo di CSS nella realizzazioni di pagine WEB sono noti: possibilità di raggruppare oggetti con stesse caratteristiche, ereditarietà ed override di attributi, creazione di nuovi stili e possibilità di riformattare velocemente l'intero sito semplicemente variando le caratteristiche dei CSS.

- **Tipografia.** Una sofisticata espressività tipografica consente di realizzare testi disposti lungo path arbitrari, utilizzando *pattern* (trame) personali e personalizzabili, e, allo stesso tempo, mantenendo capacità di ricerca ed indicizzazione su tale testo.

- **Backward Compatibility.** La grafica SVG può includere e richiamare immagini realizzati in grafica raster come ad esempio GIF, JPEG e PNG. Inoltre può includere anche altri file SVG.

- **Archivability/Searchability.** Il classico metodo di archiviazione di file grafici prevede che venga loro assegnato un "nome" sufficientemente espressivo e che vengano memorizzati in opportune directory, al fine di consentirne poi il reperimento in futuro. Con SVG, grazie alla presenza di informazioni all'interno della grafica stessa, tutto questo può essere ovviato, e realizzato mediante una semplice ricerca all'interno del file (ASCII) SVG .

- **Metadata.** La grafica può contenere informazioni di "metadata" come autore, titolo, data e copyright. Aggiungere metadata alla grafica spiana la strada a nuove interessanti applicazioni, senza significativi aumenti di dimensioni del file o di tempi di download. Quando la grafica è esprimibile tramite testo, possono essere costruiti degli indici sui componenti grafici stessi. Un uso appropriato di strutture di metadata, come ad esempio, Resource Description Framework (cfr. RDF), consente la creazione di tali indici per l'individuazione di "cosa l'immagine contiene".

- **Time and Space Efficiency.** In certi casi, dopo una compressione, la dimesione di un file di grafica vettoriale risulta considerevolmente più bassa di quella del raster equivalente. Inoltre SVG è stato pensato per raggiungere alti livelli di compressione anche utilizzando schemi come ZIP o similari, che possono poi essere decompressi "on th fly" da HTTP/1.1 .

- **"Markup Once, Draw Everywhere".** Grazie al supporto di stylesheet, le immagini in SVG possono essere ri-stilizzate (ovvero si possono modificare le loro proprietà) a seconda del contesto e dei dispositivi utilizzati (video, stampanti, sintetizzatori vocali …) … nella più amplia accezione di "look-and-feel".

- **Accessibility.** La grafica SVG è accessibile anche a utenti non vedenti, grazie a possibilità di rappresentazione alternative.

- **Internationalization.** SVG fornisce un supporto alla internazionalizzazione, per mezzo del supporto Unicode. Sfruttando il DOM Level 0 (comunemente conosciuto come e "Dynamic HTML") ogni informazione contenuta all'interno della grafica, e specifica di un particolare linguaggio (come ad esempio il titolo), può essere tradotta on the fly e servita all'utente.

- GIS e CARTOGRAFIA

Fino a qualche anno fa, gli standard per la rappresentazione di riproduzione cartografiche su Web, erano basati esclusivamente sull'usto di testo e di immagini in grafica raster. La cartografia, ovviamente, necessitava di tantissime altre funzioni complementari, che venivano realizzate mediante server script o pesanti e complicate funzioni javascript, e che, spesso, richiedevano più caricamenti consecutivi e ridondanti della stessa immagine.

A differenza della grafica raster, invece, una grafica vettoriale consente una maggior interattività con l'utente finale (basti anche solo pensare alle funzioni di zoom e panning eseguibili direttamente client-side).

La generazione di mappe è di certo una delle più importanti applicazioni di SVG. In passato, come dicevamo, è stata utilizzata anche la grafica raster per la distribuzione di mappe su web, ma, i vantaggi dell'uso di SVG sono, ora più che mai, evidenti: si pensi ad esempio alla possibilità di zoomare sui dettagli e ridimensionare la zona di interesse, senza perdere qualità dell'immagine. Inoltre, l'utilizzo di metadati, consente la realizzazione di diversi layer (es: strade statali, autostrade, caratteristiche geografiche, ecc…) che possono essere sovrapposti selettivamente per la visualizzazione. (Una mappa, per sua natura, non è altro che il risultato di una sovrapposizione di strati, ciascuno descrivente una particolare caratteristiche di una zona).

Rientrano in tale settore anche

- **Engineering.** Disegni tecnici di progettazione, ad esempio di circuiti elettronici, di motori, turbine,… per loro natura estremamente ricchi di dettagli. Linguaggi basati su XML consentono di attribuire semantica ad ogni componente del progetto; ad esempio XML-based markup language for the automotive industry costituisce un linguaggio specificamente rivolto al settore automobilistico.

- **Human Physiology.** Schemi di complessi sistemi fisiologici come il sistema nervoso o quello muscolare, necessitano di descrizioni minuziose per essere ben rappresentati. La realizzazione mediante grafica raster richiede una rielaborare dell'immagine ottenuta, per aggiungere annotazioni e richiami, mentre SVG, supportando descrizioni testuali e collegamenti ipertestuali, consente di progettare il tutto in un'unica stesura.

- **Molecular Organic Chemistry.** Esistono diverse strutture molecolari (2D) di elevata complessità (es: catene di amino acidi). I dati possono essere forniti, a seconda della natura del loro contenuto (statica o dinamica), mediante Chemical Markup Language (CML) o Molecular Dynamics Language (MoDL), rispettivamente, e la rappresentazione grafica, realizzata con SVG.

SVG permette di esprimere gli elevati contenuti informativi di una mappa, per poi visualizzarli con grafica di alta qualità. Ciò nonostante a causa della complessità dei dati geografici (proiezioni, sistemi di coordinate, , ecc.) SVG non dovrebbe essere visto come un'alternativa ai comuni formati GIS; tuttalpiù dovrebbe essere inquadrato come uno strumento di rappresentazione di informazioni geografiche, utilizzabile su web.

SVG : news

Mobile SVG Profiles: SVG Tiny and SVG Basic
[February 15,2002] W3C Working Draft

Il documento definisce due nuovi profili di SVG 1.1. Il primo, SVG Tiny, progettato esplicitamente per telefoni cellulari, ed il secondo, SVG Basic, realizzato per PDAs (Personal Digital Assistant).
Un singolo profilo non sarebbe stato sufficiente per trattare adeguatamente con tutti questi piccoli dispositivi mobili, a causa delle differenza in termini di velocità di CPU, spazio di memoria, e quantità di colori utilizzabili. Per questo si sono sviluppati

- Tiny: rivolto ai processori con le maggiori restrizioni
- Basic: per quelli con maggiori risorse

Ovviamente entrambi tengono conto delle limitate disponibilità di calcolo e di visualizzazione

Specifiche:

1. I due profili sono stati progettati per consentire a SVG di visualizzare grafica su dispositivi con tali limitazioni.
2. Entrambi massimizzano, per quanto possibile, la compatibilità con SVG 1.0 in termini di rendering dei documenti già esistenti.
3. Tali profili sono stati realizzati per facilitare le operazioni di export da altri tools, e per far sì che le transcodifiche da SVG1.1 a SVGB e SVGT preservino la massima scalabilità.

SVGT e SVGB consistono di una serie di moduli SVG 1.1. Per ciascuno di questi moduli, un dato profilo, può contenere la versione Full, Basic, Tiny o non contenerla affatto.
Ad esempio, in riferimento al modulo Structure Module, i tre profili SVG Tiny, Basic e Full ne contengono rispettivamente le versioni :

Tiny Structure Module `svg, g, defs, desc, title, metadata, use`
Basic Structure Module `svg,g,defs,desc,title,metadata,symbol,use`
Full Structure Module `svg,g,defs,desc,title,metadata,symbol,use`

Mentre, in riferimento al modulo Filter Module

(Tiny non posside il modulo)

Basic Filter Module

```
filter, feBlend, feColorMatrix, feComponentTransfer, feComposite,
feFlood, feGaussianBlur, feImage, feMerge, feMergeNode, feOffset,
feTile, feFuncR, feFuncG, feFuncB, feFuncA
```

Full Filter Module

```
filter, feBlend, feColorMatrix, feComponentTransfer, feComposite,
feConvolveMatrix, feDiffuseLighting, feDisplacementMap, feFlood,
feGaussianBlur, feImage, feMerge, feMergeNode, feMorphology, feOffset,
feSpecularLighting, feTile, feTurbulence, feDistantLight,
fePointLight, feSpotLight, feFuncR, feFuncG, feFuncB, feFuncA
```

Articoli, Riferimenti e Risorse

- W3C Graphics Activity
- W3C SVG Web site
- SVG Implementations
- SVG W3C mailing list archives
- SVG in Mozilla Project

- [February 28, 2002] "The Visual Display of Quantitative XML." By Fabio Arciniegas A. From XML.com.

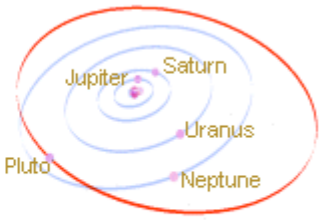
  Molto spesso i dati memorizzati in un file XML, altro non sono che enormi tabelle di numeri e riferimenti, che danno una descrizione quantitativa di un fenomeno. Casi tipici sono, ad esempio:

  - Numero e tipo di connessioni effettuati su un server (logfile);
  - Descrizione di incassi e vendite di un'attività;
  - Dati atmosferici (umidità, precipitazioni, temperatura, …) misurati in varie località e in diversi periodi;

  Questo tipo di dato ("quantitativo" appunto) è tipicamente quello in cui ci si imbatte con più frequenza; più difficili da trovare, invece, sono i modi per visualizzare con efficacia queste informazioni. La maggior parte dei riferimenti e degli articoli reperibili su rete, mostrano di solito come convertire tramite l'utilizzo di XSLT, tali dati XML in una pagina HTML, formato che, però, non è sufficientemente espressivo quando si devono rappresentare informazioni molto complesse o estremamente voluminose. I benefici e l'immediatezza di una buona rappresentazione sono evidenti:

Planet	a	e	i	Φ	~Φ	L
Jupiter	5.203	0.04839	1.30530	100.556	14.753	34.404
Saturn	9.537	0.05415	2.48446	113.715	92.431	49.944
Uranus	19.191	0.04716	0.76986	74.229	170.964	313.232
Neptune	30.068	0.00858	1.76917	131.721	44.971	304.880
Pluto	39.4816	0.24880	17.14175	110.303	224.066	238.928

e eccentricity, a semi-major axis, i inclination, Φ longitude of ascending node, ~Φ longitude of perihelion, L mean longitude



In questo articolo si discute, appunto, la creazione di una valida rappresentazione grafica di dati "quantitativi" XML utilizzando SVG.

L'esempio trattato nell'articolo di Arciniegas, descrive come realizzare una mappa dei locali (pub, ristoranti, bar, ..) di una città, a partire dai dati XML memorizzati in questo modo:

```
…
<attraction type="bar">
        <name>Moe's</name>
        <abbrev_address>
                <number>2900</number>
                <street quad="NW">B</street>
        </abbrev_address>
        <average_rating>1</average_rating>
        <average_occupants>4</average_occupants>
</attraction>
…
```

Viene disegnata una griglia identificante le varie strade della città, ed ogni incrocio viene etichettato con una coppia di label opportune. Tramite l'applicazione di template XSL, vengono scandite ad una ad una tutte le attractions, e per ciascuna viene disegnata un'icona (png) indicante il tipo di locale, in corrispondenza del suo indirizzo sulla cartina. Tale icona, poi, viene circondata da un piccolo cerchietto, il cui colore rispecchia il "grado di popolarità" (ovvero `<average_rating>`). Il tocco finale, viene dato da un piccolo script JavaScript che, sfruttando gli attributi `onmouseover` e `onmouseout` dell'elemento `<image>`, consente di far apparire una piccola descrizione di ogni locale, semplicemente quando il cursore del mouse vi passa sopra, e di farla sparire quando ci si sposta.
[*Veramente un bell'articolo !*]

References

[1] The Visual Display of Quantitative Information Edward R. Tufte Graphics Press; 2nd Edition 07/2001
[2] Planet Orbit's representation from the Exoflight simulator, courtesy of Steven Hugg
[3] The Elements of Graphing Data By William S. Cleveland CRC Press, LLC; 03/1994;
[4] Envisioning Information Edward R. Tufte; Graphics Press; 10/1990
[5] Design, Form, and Chaos Pauld Rand; Yale University Press; 04 1993
[6] Design Dialogs Steven Heller, Elinor Pettit; 09 1998
[7] What is XLink? Fabio Arciniegas, xml.com 11 2000
[8] SVG 1.0 W3C Recommendation W3C 09 2001
[9] XPath 1.0 W3C Recommendation W3C 11 1999

- [February 28, 2002] "Server Side SVG." By J. David Eisenberg. From XML.com.
  In questo articolo si utilizza il pacchetto Apache Batik project per realizzare un'applicazione Servlet che riceve dal Client la richiesta di creare un'immagine, e produce in uscita un file SVG, JPG o PNG a seconda delle caratteristiche del visualizzatore installato sul Client stesso. Batik è definito un kit basato su tecnologia Java per lo sviluppo di applicazioni che intendono visualizzare, generare e manipolare immagini in formato SVG. Batik fornisce un "trascoder" che consente di codificare una qualsiasi Stringa SVG in immagine Jpg o Png, così, una volta terminata la fase di elaborazione (che produce sempre come risultato una stringa SVG) si effettua, se necessario, una codifica nel formato richiesto dal Client.
  [*Argomento interessante, articolo un po' troppo sintetico. L'utilizzo di Batik, anche se in semplici applicazioni dimostrative, richiede comunque una certa dimestichezza con classi e metodi del pacchetto*].

- [February 28, 2002] "Doing That Drag Thang." By Antoine Quint. From XML.com
  Questo breve articolo spiega come utilizzare linguaggi di scripting assieme agli oggetti SVG, che consentono di creare vere e proprie presentazioni interattive. Essendo un'applicazione XML, SVG annovera tra le proprie caratteristiche, anche lo standard Document Object Model. DOM è costituito da un set di API object-oriented, specifiche per la lettura e la scrittura di documenti XML. DHTML è stato ottenuto per fusione di HTML, Javascript, CSS e DOM, e ciò che ha complicato l'esistenza a DHTML è stato il fatto che i due principali browser in commercio avessero due differenti standard DOM, nessuno dei due compatibile con il DOM specificato dal W3C. Recenti versioni dei browser più diffusi, supportano il W3C DOM Level 2, proprio come Adobe SVG Viewer, che, per altro, supporta anche il SVG DOM.

  Nella stesura del codice SVG è possibile inserire un riferimento ad un file contenente il codice di scripting (es. codice JavaScript), e poi utilizzarne i vari metodi nel corpo del file, in corrispondenza a particolare eventi. … si perché SVG fornisce una serie di event-listeners che catturano l'evento e ne consentono l'opportuna gestione. Esempi di eventi sono: `mousemove` (a seguito di uno spostamento del mouse), `mousedown` (in corrispondenza alla pressione di un bottone), `mouseup` (che si verifica quando il bottone viene rilasciato), …

  L'istruzione SVG : <g id="target" onmousedown="initDrag()">
  Definisce un oggetto <g> con identificatore "target" che, se premuto, scatena l'esecuzione di initDrag()

- [November 29, 2001] "SVG: Where Are We Now?" By Antoine Quint. From XML.com.
  Il 4 Settembre 2001, W3C pubblica la Reccomendation Scalable Vector Graphics 1.0, a distanza di quasi 2 anni e mezzo dalla prima public draft of February 1999. In tutto questo tempo sono state riempite più di 600 pagine di complesse specifiche, estremamente dettagliate. In questo articolo vengono proposti i vari tool SVG, quali, ad esempio, visualizzatori e editor.

  Il ***visualizzatore*** più diffuso ed avanzato è senza dubbio Adobe SVG Viewer; nella sua attuale versione (3.0) è disponibile sia per piattaforme Microsoft che Macintosh e funziona sulla maggior parte dei browser. Adobe offre anche un plug-in per "Real Player" che consente di visualizzare pagine SVG in un contesto multimediale.Un altro visualizzatore "stand-alone" SVG disponibile è Batik: progetto Java-based open source. Componente integrante del progetto XML di Apache, Batik supporta tutte le specifiche di SVG 1.0. Batik è più di un semplice visualizzatore per SVG; in realtà infatti è un vero e proprio toolkit che consente:
  - ✓ conversioni SVG-raster
  - ✓ generazione server-side di codice SVG attraverso l'uso di DOM
  - ✓ conversione di font
  - ✓ utility di stampa

  Tutte queste interessanti caratteristiche rendono Batik uno strumento prezioso per la realizzazione di applicazioni Java-based in SVG. Batik è disponibile per qualunque piattaforma Java.
  Inoltre vi sono una serie di supporti per ***browser***, trai quali vengono citati: SVG-enabled Mozilla, W3C's Amaya e X-Smiles .
  Come creare un documento SVG? Scrivere a mano il proprio codice SVG, può essere meno complicato di quanto sembri; la specification SVG è un documento leggibile, e vi sono diversi tutorial reperibili in rete, l'utilizzo dei filtri, ad esempio, può risultare sicuramente molto complicato, ma se ci si "accontenta" di opacity e gradienti, si possono ottenere splendidi risultati con poca fatica (specialmente se si ha una certa dimestichezza con XML). Stendere manualmente il codice SVG, ovviamente, consente di avere il pieno ed assoluto controllo di ogni componente grafico ed ogni effetto applicato.
  Il popolare package XML-Spy 4.1 offre un interessante supporto per SVG: dalla colorazione sintattica delle parole chiavi, al completamento automatico dei valori di elementi ed attributi, per terminare con la validazione tramite DTD (o *Schema* quando diverrà disponibile). Altra interessante caratteristica è la possibilità di creare un documento SVG a partire da una pagina XML, eseguendo una trasformazione XSL, e visualizzando direttamente il risultato sempre in una finestra di XML-Spy, esattamente come accadrebbe in un browser. Tra gli altri editor: TextPad e *vim*.
  [*Propone una breve carrellata dei visualizzatori ed editor e package SVG in circolazione (a fine del 2001)*]

- [Novembre 21,2001] "The Art is in the Code" By Eddie Traversa. From XML.com. Realizzare grafica in SVG, concettualmente, non è molto diverso dal progettare un applicazione in un qualunque linguaggio di programmazione. In questo articolo, SVG viene definito un "ponte" tra il mondo della progettazione grafica e quello della programmazione. Per altro, essendo SVG uno standard open source, è possibile approfondire le proprie conoscenze anche semplicemente cercando lavori di altri sviluppatori e studiandone il codice.
  SVG può essere definito in un documento Web in 3 modi:
  1. ***Pagina 'stand-alone'***
     Ad esempio:

```
1. <?xml version="1.0" standalone="no"?>
2. <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
     "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
3. <svg width="300" height="300" x="0" y="0">
4. <!--Your SVG content goes here -->
5. </svg>
```

     Questo metodo ha qualche limitazione, in particolare per quanto riguarda la reperibilità della pagina da parte dei motori di ricerca, la maggior parte dei quali ignorerà questo genere di pagine! L'uso

2. ***Elemento embedded***

E' possibile inserire del codice SVG all'interno di un documento HTML/XHTML come mostrato di seguito:

```
1.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
2.  <html>
3.     <head>
4.            <title>Object and Embed</title>
5.     </head>
6.     <body>
7.       <object data="test.svg" width="500" height="500"
         type="image/svg+xml">
8.            <embed src="test.svg" width="500" height="500"
              type="image/svg+xml" />
9.       </object>
10.    </body>
11. </html>
```

Per correttezza dovremmo usare solo il tag `object`, ma questo non consentirebbe la visualizzazione del sorgente SVG nei browser Netscape. E quindi è sempre meglio utilizzare entrambi i `object` e `embed`. Si noti come, `per specificare l'URL del documento SVG`, il primo tag utilizzi la proprietà `data, mentre il secondo usi src.`

Uno dei vantaggi principali dell'utilizzo di questo metodo, deriva dalla possibilità di combinare le caratteristiche di HTML e XHTML con quelle di SVG. Il nostro documento potrà quindi essere integrato con effetti sonori e sottofondi musicali, e, ovviamente, i vari motori di ricerca per il Web recupereranno tale pagina.

3. ***Utilizzato in un documento XHTML con la dichiarazione di un namespace*** (cenno)

Un altro modo di far visualizzare SVG da un browser è quello di passare attraverso l'uso di un namespace XML. Questo, che sicuramente risulta esser il più potente e flessibile dei tre metodi, è anche quello che richiede una maggiore conoscenza di base per poter essere utilizzato.

[*Articolo interessante con parecchi esempi*]

- [24 Gennaio, 2001] "What is RDF ?" By Tim Bray. From XML.com.
  RDF è l'acronimo di Resource Description Framework, ed è un'applicazione di XML. In questo articolo se ne mostrano le caratteristiche principali, analizzando tre diversi scenari, tre diverse situazioni di vita quotidiana, in cui si necessita di reperire *efficientemente* alcune informazioni.

  Scenario 1: La Biblioteca

  Vogliamo trovare libri che parlino di un certo argomento. In molte librerie, questa ricerca è effettuata tramite l'uso di un computer, che non fa altro che gestire un'elenco di record (uno per ogni volume) esattamente come accadeva qualche anno fa con i raccoglitori cartacei. Questo meccanismo consente di elencare per autore, titolo, argomento, ecc.. , e la lista comprende, oltre a molte informazioni di ogni singola pubblicazione, anche (forse più importante di tutte) il riferimento allo scaffale nel quale poter trovare il libro.

  Scenario 2: Il Video Noleggio

  Ci troviamo in un negozio di video cassette e stiamo cercando un film di John Huston. La maggior parte dei video noleggi offre la possibilità di effettuare le nostre ricerche tramite l'uso di un computer, esattamente come accadeva nell'esempio precedente. Ovviamente i parametri di ricerca non saranno gli stessi, ma il risultato, più o meno, non sarà molto diverso.

Scenario 3: L'elenco telefonico

Vogliamo ordinare una pizza, e non conosciamo il numero di telefono della pizzeria più vicina. Consultiamo le pagine gialle…

Ciò che accomuna tutte e tre queste situazioni è, innanzi tutto, la necessità di *metadati* ovvero di informazioni che si riferiscono ad informazioni. In ciascun esempio, infatti, si ha bisogno di un dato (lo scaffale del libro, il nome del video, il numero di telefono della pizzeria), e per reperirlo usiamo appunto dei metadati.
Punto chiave di tutta la discussione è che: in teoria, i metadati non sono affatto necessari …
… potremmo entrare in libreria (o nel video noleggio) e controllare i libri (o le video cassette) uno ad uno, finchè non troviamo quello che ci interessa, oppure comporre tutti i numeri della zona, finchè non troviamo quello di un pizzeria …
… ma tutto ciò, oltre ad essere inattuabile, sarebbe anche abbastanza stupido. Ecco, i metadati servono appunto a questo!In ciascun esempio utilizziamo queste meta-informazioni per raggiungere il nostro scopo, e lo facciamo seguendo sempre la stessa tipologia di ragionamento, e ciò, nonostante i vari metadati siano completamente differenti da caso a caso. Questo, però, non ci turba minimamente, forti del fatto che ci riteniamo in grado di applicare le informazioni giuste nella situazione opportuna.

Per di più vi sono un'infinità di altre informazioni che l'utente finale non vede direttamente e che non utilizza nelle proprie ricerche, ma che sono fondamentali per il mantenimento e l'erogazione dei servizi della libreria, del negozio di video cassette, e dell'elenco telefonico (ad esempio: la frequenza con cui un libro o una cassetta vengono richiesti, il loro costo originale, il fornitore, e così via).

Il Web, a sua volta, è pensabile come un'immensa libreria con milioni di documenti; solo conoscendone l'URL (il numero di telefono a tutti gli effetti) è possibile reperili. Il Web, però, contiene sia libri, che film, che pizzerie e i meccanismi con i quali si possono ricercare le informazioni comprendono quelli specifici delle librerie, più quelli dei video noleggi, più quelli dell'elenco telefonico, e tanti altri ancora. Viene spontaneo chiedersi in che modo poter reperire documenti su Web. Motori di ricerca come Altavista, Infoseek ed Excite usano "la forza bruta", ovvero è come se, tramite i propri robot di ricerca, entrassero in ogni libreria, leggessero tutti i libri e ce ne riportassero gli argomenti in base alle parole contenute nelle loro pagine. Yahoo! non utilizza robots, si basa invece sulla presenza di etichette associate ai siti, e liste di categorie, che ricoprono a tutti gli effetti il ruolo di metadati.

Esiste uno standard unico di metadati per i documenti contenuti su Web?
Beh, sarebbe una richiesta un po' pretenziosa: le stesse librerie, dopo tanti anni, continuano ad avere ciascuna un formato diverso per la descrizione delle proprie meta-informazioni …

Nei tre esempi iniziali, però, si è mostrato come vi siano delle caratteristiche comuni a metadati di provenienza diversa; RDF rappresenta il tentativo di identificare questi aspetti comuni, per consentire una strutturazione del Web, che consenta il reperimento *efficiente* delle informazioni in esso contenute.

RDF (Resource Description Framework) è un framework per la descrizione e lo scambio di metadata. Concetti base di RDF sono:

1. Una **Risorsa** è tutto ciò che possiede un URI (Uniform Resource Identifier); tutte le pagine Web, così come tutti gli elementi di un documento XML;
2. Una **Proprietà** è una risorsa che possiede un nome, e che può essere utilizzata come attributo nelle ricerche, ad esempio `Autore` o `Titolo`. Il più delle volte ci interessiamo solo del nome, ciò nonostante, il fatto di essere una risorsa, permette alla proprietà di avere a sua volta delle proprietà.;
3. Uno **Statement** consiste nella combinazione di Risorse, proprietà e valori, rispettivamente ('soggetto', 'predicato', e 'complemento' dello Statement). Ad esempio:
   " The Author of `http://www.textuality.com/RDF/Why.html` is Tim Bray."
   Il valore può essere una stringa (come nell'esempio appena visto) o un'altra risorsa:
   "The Home-Page of `http://www.textuality.com/RDF/Why.html` is

4. In XML c'è un modo per esprimere tutto questo:

```
<rdf:Description about='http://www.textuality.com/RDF/Why-
RDF.html'>
<Author>Tim Bray</Author>
<Home-Page rdf:resource='http://www.textuality.com' />
</rdf:Description>
```

RDF è stato ideato per avere le seguenti caratteristiche:

- ✓ **Indipendenza**
  Una proprietà è una risorsa, e quindi ognuno può inventare nuove proprietà.
- ✓ **Scambiabilità**
  Gli statement RDF possono essere convertiti in XML, e questo li rende facilmente condivisibili con altri.
- ✓ **Scalabilità**
  Il Web sta ingrandendosi sempre di più, per uno standard che deve ricercare documenti fra bilioni di candidati, la scalabilità è un requisito fondamentale.
- ✓ **Le Proprietà sono Risorse**
  Le Proprietà possono avere a loro volta delle proprietà, e possono essere ricercate e manipolate come tutte le altre risorse. Questo, visto il loro gran numero, è molto importante; ad esempio potrei voler verificare se qualcuno ha già definito una proprietà che descrive il genere di un film con valori tipo Commedia, Horror, Thriller, Comico. Per farlo avrò bisogno di metadata!
- ✓ **I Valori possono essere Risorse**
  Ad esempio molte pagine Web avranno la proprietà "Home-Page" che punta all'home page del proprio sito.
- ✓ **Gli Statement possono essere Risorse**
  Anche gli statement possono avere delle proprietà. Yahoo! funziona tramite metadata forniti da altri, e quindi è legittimo, dato lo statement "Questa pagina ha come argomento XXX", chiedere "chi ha dato questa informazione, e quando?".I metadata rappresentati dalle proprietà di uno statement rispondono a questo genere di domande.

L'uso di RDF

- Consente alla macchina di "capire" la semantica associata ai dati (tramite l'uso dei metadati)
- Consente una migliore precisione nel reperimento di documenti, di quanto non lo sia la semplice ricerca di parole chiavi contenute nel testo

E perché non utilizzare semplicemente XML? XML consente di creare nuovi tag, che possono a loro volta contenere semplice testo o altri tag. Ciò che però ostacola l'utilizzo di XML per lo scambio di metadata general porpouse è la scalabilità. Esistono due problematiche:

1. L'ordine in cui appaiono gli elementi in XML è rilevante; mentre non ha interesse, durante la ricerca del solito film dell'esempio, che l'autore compaia prima del titolo. In oltre mantenere ordinate le proprietà di milioni di oggetti sarebbe pressochè impossibile.
2. XML consente costruzioni del tipo:

```
<Description>The value of this property contains some
text, mixed up with child properties such as its
temperature(<Temp>48</Temp>) and longitude
(<Longt>101</Longt>). [&Disclaimer;]</Description>
```

La rappresentazione interna di un documento XML mischia assieme alberi, grafi e stringhe di testo, il che la rende difficile da gestire anche per piccole modeste quantità di oggetti (figuriamoci per i milioni di milioni di oggetti presenti su Web).

D'altro canto XML è quanto di meglio si possa desiderare per la "scambiabilità" delle informazioni, e già per questo merita di far parte del patrimonio di RDF (ma da solo, XML, non basta per realizzare un framework di metadati).
RDF fornisce un modello e una sintassi per la gestione dei metadati; ciò che non fornisce, invece, è la

Esistono dei package chiamati **Vocabolari**, contenenti le proprietà a descrizione di un'enorme quantità di argomenti diversi: libri, video, vini pregiati, fondi d'investimento, e tanti altri ancora.
Ognuno può inventare, proporre e vendere un proprio vocabolario, ed ogni singolo utente può decidere se utilizzarlo o meno; con una sorta di selezione naturale, i "migliori" di questi vocabolari (quelli più accettati o utilizzati) sopravviveranno alla concorrenza, e continueranno ad esistere; gli altri spariranno gradualmente.
RDF è diventato Reccomendation W3C nel Febbraio 1999.

Come inserire RDF in una pagina HTML?
Il metodo più veloce è quello di inserire all'interno del tag <head> qualcosa come:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://documento"
      dc:creator="Joe Smith"
      dc:title="My document"
      dc:description="Joe's ramblings about his summer vacation."
      dc:date="1999-09-10" />
</rdf:RDF>
```

dove

`http://doc` è l'URL della pagina

creator, title, description e date : rappresentano le proprietà della pagina.

Per ulteriori informazioni sui namespace si veda " Dublin Core Metadata Element Set: Reference Description".

SVG Specification

- [15 Febbraio, 2002] SVG 1.1 Working Draft.
  "Questa versione di SVG fornisce una modularizzazione di SVG release 1.0, che può essere sfruttata per creare profili di SVG come ad esempio 'SVG Mobile'. (...) Questi moduli possono essere combinati assieme per creare subset ed estensioni di SVG. (...). Ogni sezione principale di SVG ha un omonimo modulo a cui è anteposto uno di questi tre prefissi : Full, Basic e Tiny, a seconda del sottoinsieme di elementi ed attributi che vengono inclusi nella versione.
  Un modulo Tiny è un subset del Basic, che, a sua volta, è un subset del Full."

- [8 Gennaio, 2002] SVG 1.1 Working Draft
- [30 Ottobre, 2002] SVG 1.1 Working Draft

- [04 Settembre, 2001] SVG 1.0 Specification diviene Recommendation W3C. Il World Wide Web Consortium scrive:
  "SVG rappresenta un punto di incontro tra un linguaggio XML-based e la possibilità di creare grafica vettoriale bidimensionale". [*Testo Completo*]

- [11 Agosto, 2001] W3C pubblica SVG Requirements Specification. Il Working Group W3C di SVG ha pubblicato 2 Working Draft, a specifica dei requisiti della prossima fase della progettazione di SVG.
  "SVG è un linguaggio per la definizione di grafica bidimensionale che utilizza una sintassi XML per descrivere elementi grafici, consentendone la rappresentazione in maniera indipendente dalla risoluzione. La Specification SVG 1.0, attualmente allo stato di Proposed Recommendation, discute la visualizzazione di elementi che possono essere utilizzati in applicazioni SVG autonome, o incluse in altri documenti XML. Il prossimo step nello sviluppo di SVG, sarà la realizzazioni di due specifications, SVG 1.1 e SVG 2.0. SVG 1.1 includerà una versione modulare di SVG 1.0, e nuove caratteristiche guidate dall'esigenza di un profilo di SVG per dispositivi mobili. Parallelamente al suo sviluppo, 'SVG Working Group' svilupperà un insieme di profili come full SVG, SVG Tiny e SVG Basic. SVG 2.0 includerà le caratteristiche di SVG 1.1, aggiungendone di nuove." [*Testo Completo*]

- [21 Luglio, 2001] W3C pubblica SVG1.0 Proposed Recommendation (*Scalable Vector Graphics (SVG) 1.0 Specification*). Il documento definisce le caratteristiche sintattiche di SVG, linguaggio per la descrizione di vettori bidimensionali, e grafica mista vettoriale/raster in XML. La grafica vettoriale SVG è scalabile a qualsiasi livello di risoluzione. La maggior parte delle grammatiche XML rappresentano informazioni testuali o comunque dati grezzi, e tipicamente forniscono rudimentali capacità di espressione grafica. SVG riempe il gap fornendo una ricca descrizione grafica degli elementi, utilizzabile sia in applicazioni SVG stand-alone, che come namespace XML con altre grammatiche. [*Testo Completo*]

- [03 Novembre, 2000] W3C SVG Working Group inoltra una Candidate Recommendation (riveduta) per *Scalable Vector Graphics (SVG) 1.0 Specification*, parte integrante del W3C Graphics Activity. Questa Recommendation definisce le caratteristiche e la sintassi di SVG:
  "un linguaggio per la descrizione di vettori bidimensionali, e grafica mista vettoriale/raster in XML. SVG prevede l'utilizzo di 3 tipi di oggetti grafici: forme (cammini composte da linee e curve), immagini e testo. Tali oggetti possono poi essere ragggruppati e vi si possono applicare stili, trasformazioni e filtri. Si possono creare animazioni sia dichiarativamente (inserendone il codice direttamente nel contesto SVG) o via scripting, utilizzando linguaggi di scripting esterni per accedere al DOM del documento SVG, che fornisce la lista completa di

- [02 Agosto, 2000] Il W3C SVG Working Group, inoltra *Scalable Vector Graphics (SVG) 1.0 Specification* come Candidate Recommendation, e parte integrante del W3C Graphics Activity.
  Riferimenti: W3C Candidate Recommendation 02-August-2000, edited by Jon Ferraiolo (Adobe).
  Il periodo di revisione di una Candidate Recommendation termina quando almeno una implementazione di SVG supera tutti i test denominati Basti Effectivity (BE) test contenuti nella SVG test suite
  Si vedano W3C press release e 'Testimonials for Scalable Vector Graphics Candidate Recommendation.'

- [09 Agosto, 2000] *Accessibility Features of SVG*. Riferimento: W3C Note 7-August-2000, edited by Charles McCathieNevile and Marja-Riitta Koivunen.

  "Scalable Vector Graphics (SVG) dispone di un insieme di caratteristiche che possono rendere la grafica su Web più accessibile di quanto non lo sia già ora, ad un pubblico più vasto di utenti, tra cui: non vedenti o persone con limitata capacità visiva, utenti non in grado di leggere o comprendere testo, o con difficoltà nell'utilizzare il mouse o la tastiera, o semplicemente utenti che utilizzano display di solo testo, monitor di dimensioni estremamente ridotte, o molto grandi. Questo è il concetto che sta alla base del formato SVG: massimizzare il numero di utenti in grado di accedere ad una generica immagine (accessibility) non escludendo persone che, pur non avendo alcuna invalidità, hanno particolari necessità e richieste. Ad esempio persone che utilizzano tali informazioni in situazioni in cui la vista sia già impegnata su altri oggetti (es: mentre si è al volante della propria vettura, ...), o che facciano uso di piccoli dispositivi portatili (video palmari, senza tastiera o mouse)."

- [10 Luglio, 2000] Il SVG Working Group ha realizzato un Working Draft (riveduto) per *Scalable Vector Graphics (SVG) 1.0 Specification* parte integrante del W3C Graphics Activity. Riferimento: W3C Working Draft 29-June-2000, edited by Jon Ferraiolo (Adobe). Nell'Appendix A del working draft è possibile trovare SVG XML DTD.

- [09 Marzo, 2000] Last Call Working Draft per il W3C's Scalable Vector Graphics (SVG) 1.0 Specification. Riferimento : W3C Working Draft 03-March-2000, edited by Jon Ferraiolo (Adobe).

- [29 Ottobre, 1998] *Scalable Vector Graphics (SVG) Requirements*. W3C Working Draft. [WD-SVGReq-19981029] Edited by Jon Ferraiolo (Adobe Systems Incorporated). "W3C ha costituito un Scalable Vector Graphics working group, per produrre una specification per il formato SVG, realizzato come set di tag XML ed utilizzabile come namespace XML."

Conclusioni

In questa esposizione si sono evidenziate caratteristiche, modalità di impiego, vantaggi e svantaggi di questo nuovo formato. Uno standard, però, deve essere accettato non solo dalle case costruttrici di software, ma anche dal pubblico degli utenti.

Per avere il feedback degli utilizzatori, è sufficiente entrare in qualche gruppo di discussione o visitare qualche sito web dedicato all'argomento. Ciò che si percepisce immediatamente, nel caso di SVG, è da un lato l'innegabile interesse che questo nuovo formato suscita nei lettori, ma dall'altro anche il timore che si ha nell'addentrarsi in una tecnologia tanto potente quanto complicata.

Numerosi sono i tutorial e le guide che si possono reperire su rete, ma ciò che frena il grande pubblico davanti alla tentazione di fare il primo passo verso il nuovo formato, è inevitabilmente la consapevolezza che SVG, così come DTHML del resto, sono tecnologie che, per essere padroneggiate, richiedono capacità e conoscenze superiori a quelle possedute dalla figura dello sviluppatore medio.

SVG rappresenta un grande passo avanti nella storia dell'Infomatica teorica, ma al contempo fatica ad attechire nella realtà quotidiana.

Di fatto esistono già centinaia di formati per la progettistica (CAD), per la cartografia (GIS), per la grafica e l'animazione su Web, e tutti funzionano bene. Cosa può spingere l'utente medio di queste tecnologie a cambiare la propria radicata fiducia in uno di questi prodotti e impiegare tempo e risorse per dedicarsi allo studio di un nuovo formato? Forse il 'miraggio' della standardizzazione del Web?

Sembra un'utopia, ma è proprio questa la chiave del successo di SVG !
Adottare SVG come forma intermedia di descrizione dei propri dati, spianerà la strada all'utilizzo di un'infinità di risorse: partendo dai pacchetti realizzati (e di futura realizzazione) SVG compatibili, fino ad arrivare alla possibilità di scambio di dati fra applicazioni diverse, con specifiche diverse, e realizzati da produttori diversi, che condividono il formato di rappresentazione delle informazioni.

Siamo ora in grado di rispondere alla domanda con la quale si era aperta la fase introduttiva di questa relazione…

Il fatto di essere un nuovo formato di grafica vettoriale, rappresenta, forse, solo la punta dell’iceberg per SVG; le potenzialità nascoste al di sotto lo candidano come un valido standard per la descrizione, memorizzazione, manipolazione e rappresentazione di qualsiasi tipo di dato.

APPENDICE A
Legenda

Nel presente documento si troveranno diverse formattazioni del testo:

- Questo è testo normale
- *Questo rappresenta un estratto da una pubblicazione o da un articolo*
- *I commenti e le oprinioni personali, hanno questo formato*
- I link e i riferimenti a pagine Web appaiono in questo modo

```
Il codice sorgente (SVG, XML, XSL, …)
ha questo
aspetto
```

APPENDICE B
XML : Concetti di base

XML *(Extensible MarkUp Language)* è un linguaggio basato su markup del testo, e sta diventando velocemente lo standard per lo scambio di informazioni su WEB. Come HTML, XML utilizza dei TAG (identificatori), ma diversamente da quelli di HTML i TAG di XML descrivono **cosa** un dato rappresenti, e non **come** lo si debba visualizzare. Per chiarire: HTML dice "*visualizza questo in corsivo*", mentre XML dice "*questo pezzetto di dato rappresenta un numero di matricola*". Allo stesso modo in cui si è soliti definire i nomi dei campi di una struttura dati, è possibile usare un omonimo tag XML per *markare* quel campo. Questo consente di ritrovare le informazioni all'interno di un testo, ricercando il nome del field corrispondente.

Le motivazioni che hanno portato alla creazione di XML sono le seguenti :

1. XML è una delle infinite rappresentazioni esterne di un dato, e sta diventando sempre più velocemente lo standard di scambio delle informazioni su WEB. E' quindi fondante, anche, come rappresentazione interna, visto che, essendo in un formato universalmente riconosciuto ed accettato, non ci saranno problemi nello scambio con terzi.
2. XML è già nel patrimonio di ogni Browser. In particolare le ultime versioni forniscono anche la possibilità di visualizzare e modificare file XML, alla stregua di file in formato HTML.
3. XML è una rappresentazione indipendente dal linguaggio di programmazione utilizzato, quindi, uno stesso dato rappresentato in XML può essere processato da programmi implementati in Java, C, C++, Lisp, ...
4. Analizzatori lessicali (Lexer) e e sintattici (Parser) XML sono gratuitamente disponibili su WEB, e quindi non è richiesto alcuno sforzo nè concettuale, nè implementativo per processare documenti XML.

W3C (consorzio no-profit che si occupa degli standard WEB), ha poi elencato una serie di ulteriori benefici di cui godono i documenti XML, nella sezione Extensible Markup Language (XML) 1.0 W3C Recommendation.

Si elencano di seguito alcuni di questi:

1. **Plain Text**
   Non essendo XML un formato binario, è possibile creare e modificare file utilizzando dal comune text editor, al sofisticato ambiente di sviluppo visuale. Questo consente di facilitare la memorizzazione di piccole quantità di dati, così come la gestione di consistenti DataBase.
2. **Data Identification**
   XML dà la descrizione dei *tipi di dati* da cui l'informazione di partenza è costituita, e quindi delle diverse parti in cui essa è scomponibile. Per questo motivo, è l'ideale per l'identificazione e il reperimento di dati, indipendentemente dalle differenti applicazioni che ne faranno uso.
3. **Stylability**
   Quando anche la visualizzazione del dato diventa importante, XSL *(Extensible StyleSheet Language)* consente di impostare le proprietà di formattazione. Ad esempio, supponendo di avere un DB contenente i dati personali, tra cui anche l'indirizzo di posta elettronica, di una comunità di persone; un frammento di descrizione XML potrebbe essere :
   ```
   <mail_address> emanuelventuri@flashmail.com </mail_address>
   ```
   Con XSL se ne può dare una visualizzazione, specificando
   1. di iniziare una nuova linea
   2. di visualizzare "Mail Address :" in grossetto, seguito da uno spazio bianco
   3. di visualizzare il dato contenuto nell'indirizzo di posta

   Ciò che si ottiene è

   Mail Address: emanuelventuri@flashmail.com

   Ovviamente, si sarebbe potuto ottenere lo stesso risultato con HTML, con la differenza, che, però, non si sarebbe riusciti a estrapolare il contenuto del campo *mail_address*, dalla base di dati di partenza.
   Ancora più interessante, è la possibilità di usare con XML, i più svariati fogli di stile, potendo così produrre in uscita
   - postscript
   - TEX
   - PDF
   - HTML
   - SVG
   - o qualche altro formato (che non sia ancora stato inventato !)

4. **Inline Reusability**
   Uno degli aspetti più apprezzati dei documenti XML è rappresentato dalla possibilità di comporli a partire da entità separate. Anche HTML ha questa proprietà, ma la realizza, esclusivamente tramite link ad altri documenti. In XML, invece, tali entità possono essere incluse "in linea", consentendo di effettuare una ricerca sull'intero documento.
5. **Easily Processed**
   Come menzionato in precedenza, le rigide regole grammaticali di XML con cui i dati vengono descritti, consentono di creare semplici programmi in grado di processare le informazioni XML. Ad esempio, in HTML, un tag *<dt>* può essere chiuso con un tag *</dt>*,*</dd>*o *</dl>*. Ciò crea non poche difficoltà per l'utilizzo di un tale dato, dentro un programma. In XML, invece, un tag *<dt>* deve essere sempre chiuso da un tag *</dt>*. Questa restrizione consente al parser XML di identificare tutti i nodi dell'albero, senza ambiguità.
6. **Hierarchical**
   I documenti XML godono di una struttura gerarchica. Grazie a questa caratteristica consentono a chi li naviga, di saltare velocemente (in pochi step), al frammento di dato che interessa, in modo del tutto analogo a quanto accade quando si sfoglia un'indice per argomenti.
   In questo, XSL, è supportato da un altro linguaggio : XPATH, permette di accedere, a partire da un nodo, al nodo padre, ai nodi figli, ai predecessori e successori, che verifichino particolari condizioni espresse su attributi, cardinalità, posizione relativa ed assoluta.

Il linguaggio XSL, permette, inoltre, di descrivere delle regole (chiamate **Pattern**), secondo le quali un documento XML deve essere modellato. In pieno stile dichiarativo, quindi, è possibile richiedere che un certo albero XML contenga nei propri nodi foglia, la rappresentazione in stringa di una lista di parole, la valutazione di una sequenza di espressioni numeriche, e così via . Riaffiora quindi il concetto di valutazione, ed è immediata la relazione fra dati descritti in XML, e regole espresse in XSLT.

APPENDICE C:
Tipi di Software: convenzioni

Software libero (Free Software)

Il software libero è software distribuito in modo che chiunque ne abbia il permesso di uso, copia e distribuzione, in forma modificata o meno, gratis o a pagamento. Attenzione perché il termine *free* in inglese significa sia gratuito che libero, e questo, spesso, porta a fraintendimenti. Essere libero o meno, non è una questione di prezzo, ma di diritti !

Software Open Source (Open Source Software)

Il termine "open source" software è usato da alcuni più o meno con lo stesso significato di software libero. Il codice sorgente deve essere disponibile.

Software di pubblico dominio (Public Domain Software)

Il software di pubblico dominio è software privo di copyright. È un caso speciale di software libero senza permesso d'autore, il che significa che alcune copie o versioni modificate possono non essere affatto libere.

Talvolta si usa il termine "dominio pubblico" in un'accezione vaga per intendere "libero" o "disponibile gratuitamente". Tuttavia "di dominio pubblico" è un termine legale che significa precisamente "senza copyright".

Software con permesso d'autore (copyleft) (Copylefted Software)

Il software con permesso d'autore è software libero, per il quale, all'atto della ridistribuzione o della modifica, non è permesso porre alcuna restrizione addizionale. Questo significa che ogni copia del software, anche se modificata, deve essere software libero.

Software libero senza permesso d'autore (Non-Copylefted Software)

L'autore di software libero senza permesso d'autore dà il permesso di ridistribuire e modificare il programma, e anche di aggiungervi ulteriori restrizioni.

Se un programma è libero, ma non ha permesso d'autore, alcune copie o versioni modificate possono non essere libere.

Software semilibero (SemiFree Software)

Il software semilibero è software non libero, ma che nella distribuzione vede accordato il permesso ai privati di essere usato, copiato, distribuito e modificato (incluse le versioni distribuite con modifiche) senza scopo di lucro.

Software proprietario (Proprietary Software)

Il software proprietario è quello che non è nè libero, nè semilibero. Il suo utilizzo, la ridistribuzione o modifica sono proibiti o richiedono un permesso.

Software Commerciale (Commercial software)

Il software commerciale è software sviluppato da un'azienda allo scopo di guadagnare dal suo uso. "Commerciale" e "proprietario" non sono la stessa cosa! La maggior parte del software commerciale è proprietario, ma c'è software libero commerciale, e c'è software non commerciale non libero.

APPENDICE D
W3C: convenzioni

Notes A Note is a dated, public record of an idea, comment, or document. A Note does not represent commitment by W3C to pursue work related to the Note.

Working Drafts A Working Draft represents work in progress and a commitment by W3C to pursue work in this area. A Working Draft does not imply consensus by a group or W3C.

Candidate Recommendations A Candidate Recommendation is work that has received significant review from its immediate technical community. It is an explicit call to those outside of the related Working Groups or the W3C itself for implementation and technical feedback.

Proposed Recommendations A Proposed Recommendation is work that (1) represents consensus within the group that produced it and (2) has been proposed by the Director to the Advisory Committee for review.

Recommendations A Recommendation is work that represents consensus within W3C and has the Director's stamp of approval. W3C considers that the ideas or technology specified by a Recommendation are appropriate for widespread deployment and promote W3C's mission.