Produzione di documenti SVG Tiny

da GeoMedia WebMap

Caso S., Fortunati L., Massei G.

Ottobre 2005

Istituto di Scienza e Tecnologie dell'Informazione

Indice

Introduzione

La produzione di mappe da parte di Web Map Server è generalmente orientata verso due tipi di rappresentazione:

a) ***raster**:* costituita da immagini contenute in file di formato PNG, JPG, …;
b) ***vector**:* costituita da un insieme di primitive grafiche contenute in file di formato CGM, SVG, ….

La rappresentazione col formato SVG fa riferimento allo standard W3C (Raccomandazione **SVG 1.x**) per dispositivi di visualizzazione tradizionali (quali ad es. i Personal Computers), denominato **SVG Full**. W3C definisce inoltre per i dispositivi mobili (con limitate risorse di banda trasmissiva, memoria, capacità elaborativa, dimensione e caratteristiche del display) due standard **SVG (Mobile SVG**), derivati come profili della Raccomandazione SVG 1.1: essi sono specifici per i dispositivi PDA (**SVG Basic**) e per i telefoni cellulari (**SVG Tiny**). In particolare si può osservare che SVG Tiny è un sottoinsieme di SVG Basic, il quale a sua volta è un sottoinsieme di SVG Full. Ne deriva pertanto che Mobile SVG è stato progettato in modo che SVG 1.1 possa essere trascodificato in SVG Basic e SVG Tiny, mantenendo quanto più possibile la scalabilità.

In questo contesto si inquadra l'attività in oggetto, avente come obiettivo quello di rendere fruibili i dati SVG prodotti dal server GeoMedia® WebMap Professional (per i Personal Computers) anche ai dispositivi mobili, ed in particolare ai telefoni cellulari. Trattando in modo specifico l'informazione territoriale, le operazioni di trascodifica (da SVG Full a SVG Tiny) interesseranno in modo particolare l'insieme degli elementi grafici finalizzati alla visualizzazione degli oggetti geografici. Il prototipo sviluppato prevede l'implementazione sul Web Map Server Intergraph GeoMedia WebMap e la operatività su telefoni cellulari compatibili con l'ambiente Java MIDP 2.

1. Il Server *GeoMedia WebMap*

GeoMedia® WebMap Professional è un server prodotto da Intergraph Corporation per il sistema operativo Windows Server che consente di pubblicare viste di un database GIS sul World-Wide Web.

WebMap è anche un toolkit di programmazione per predisporre pagine Web con contenuti GIS ed include strumenti di analisi spaziale e di manipolazione per creare applicazioni Web che utilizzano interrogazioni spaziali, buffering e trasformazione di coordinate. Mediante questi strumenti si possono creare applicazioni specifiche di Web Mapping che sono dinamiche, aperte e scalabili.

WebMap può creare mappe sia nel formato vector (Scalable Vector Graphics (SVG) o ActiveCGM™) che raster (JPEG e PNG). Le mappe possono essere create "on the fly" a partire da sorgenti eterogenee di dati GIS, senza la necessità di conversioni di formato. WebMap ha la capacità di convertire automaticamente e combinare oggetti geografici in varie proiezioni geografiche, da sorgenti dati multiple, in una singola proiezione di mappa per la visualizzazione. Supporta inoltre in input un elevato numero di formati raster.

GeoMedia WebMap accede ai dati geografici organizzati in Wharehouse effettuando connessioni specifiche con ciascun Wharehouse. Un Wharehouse infatti contiene solo un tipo di dato geografico (Access, FRAMME, MGE, , MGESM, ARC/INFO, Oracle, ArcView, MapInfo, MGDM, or CAD). GeoMedia WebMap accede anche a GeoMedia data servers distribuiti in rete, consentendo di pubblicare così informazioni GIS da una o più sorgenti.

Ciascuna connessione ai Wharehouse è definita mediante un Map Definition File (MDF), che specifica quali *feature*[1] visualizzare nella mappa di output e le modalità con cui queste devono essere rappresentate.

2. La produzione di documenti SVG di *GeoMedia WebMap*

GeoMedia WebMap crea documenti SVG secondo le specifiche SVG 1.0, con l'aggiunta di alcuni attributi ed identificatori relativi ad elementi grafici. Il supporto SVG di WebMap è limitato alla rappresentazione del modello ad oggetti e di programmazione di WebMap.

Il processo di produzione del documento SVG avviene in due passi:

[1] In GeoMedia, una *feature* è una entità geografica rappresentata su una mappa dalla geometria e definita da attributi non grafici in un database. Sono definite: *Point Features*, *Linear features*, *Area features*, *Text features*.
Una *feature class* corrisponde alla classificazione a cui ciascuna istanza della feature è assegnata.
Un *Feature Set* contiene le *feature class* del database che si vogliono rappresentare con la stessa simbologia. Questa entità è associata anche a quelle di *Sheet* o *Layer*, intese come insiemi di *feature* vector.
Una *Feature View* definisce come uno o più *Feature Set* dovranno essere rappresentati e le relative regole di rappresentazione.

1) WebMap trasforma i dati di base, la geometria e le impostazioni del ***MapServer*** in un formato SVG "di base", aggiungendo alcuni attributi ed identificatori relativi ad elementi grafici. Nessuna trasformazione XSLT è eseguita. Può quindi essere prodotto un documento SVG con questo formato, facendo uso della proprietà **SVGFileSaveMode** di MapServer (il file prodotto ha estensione ***.SVG.XML***).
2) se un file XSL è specificato, viene successivamente eseguita la trasformazione XSLT mediante il Microsoft MSXML Parser. Facendo uso del file XSL proprio di WebMap (***GWM.XSL***), viene aggiunta l'interattività al file SVG prodotto, mediante l'inserimento di elementi grafici SVG e link a file JavaScript (il file prodotto ha estensione ***.SVG***). I file JavaScript forniscono una API per interagire col contenuto SVG e processare eventi SVG MapServer.

3. Gli elementi grafici SVG generati da *GeoMedia WebMap*

GeoMedia WebMap produce documenti SVG "di base" rappresentando le varie entità geografiche di GeoMedia con primitive SVG (vedi tabella). Da notare che SVG non prevede la primitiva punto, pertanto la feature geografica puntuale viene trasformata in un rettangolo (dagli spigoli smussati).

Primitiva GeoMedia	***Primitiva SVG***
Punto	Rect
Linea	Path
Area	Path (chiuso)

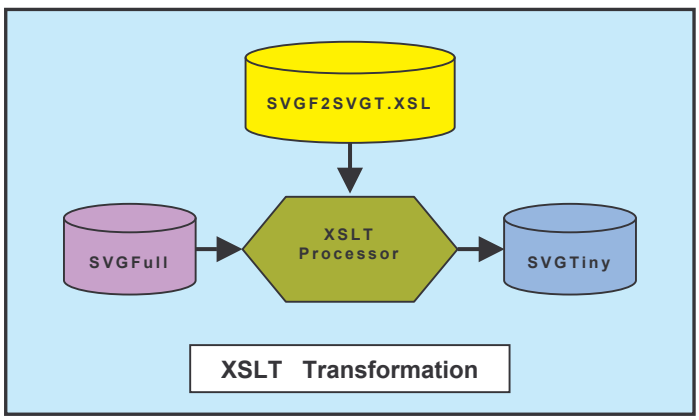
Per la rappresentazione di feature lineari e areali SVG offre le primitive <polyline> e <polygon>, rappresentanti tutte le coordinate dei punti della linea o del bordo areale. In alternativa, la primitiva <path> descrive la forma della geometria tramite coordinate e comandi; i comandi (caratteri maiuscoli o minuscoli a seconda che si faccia riferimento ad un posizionamento assoluto o relativo) definiscono come rappresentare le linee: un poligono quindi è rappresentato con un *path* chiuso utilizzando il comando "Z". L'esempio riportato sotto, mostra l'impiego della primitiva *path* per rappresentare un'area ed una linea da parte di GeoMedia WebMap.

```
<path id = "area" d = "M250 150 L150 350 L350 350 Z" />
<path id = "linea" d = "M250 150 L150 350 L350 350" />
```

Nel documento SVG è poi presente l'elemento <g> per rappresentare Feature Set di GeoMedia. Questo elemento può contenere altri elementi SVG eterogenei, e consente di rappresentare sia sottoinsiemi di uno o più strati informativi geografici che strati informativi geografici molteplici, o combinazioni di entrambi i casi.
Quindi in generale non si può dire come una mappa (intesa come composizione di più strati informativi vettoriali) può essere rappresentata da elementi SVG: questo dipende essenzialmente dalle caratteristiche di rappresentazione (styling), in quanto rappresentazioni omogenee (anche di elementi geografici eterogenei) vengono rappresentate mediante elementi <g> di SVG.
Come WebMap rappresenta un layer raster.

4. La trasformazione XSL da SVG Full a SVG Tiny

Nell'operare la trasformazione che deve portare un documento in formato SVG Full ad essere fruibile da un dispositivo mobile, si devono prendere in esame le caratteristiche che distinguono SVG Tiny da SVG Full e trasformare in maniera opportuna gli elementi che devono essere visualizzati.
In questo processo rientrano anche le "semplificazioni" che eliminano alcuni elementi "superflui" (sia perché non specifici di un contesto *GIS-oriented*, sia perché trattasi di elementi non supportati, quali ad esempio le funzioni di scripting) e riducono ulteriormente la dimensione del documento. Quindi il processo di trasformazione ha una duplice valenza: quella di compatibilità con SVG Tiny e quella di riduzione delle dimensioni del documento SVG. In questo processo risulterà comunque inevitabile una perdita di funzionalità, previste in SVG Full e non riproducibili in SVG Tiny.



Il processo di trasformazione avviene avvalendosi di un processore XSLT che:

- accetta in ingresso un documento in formato SVG Full;
- applica le regole di trasformazione contenute in un *foglio di stile XSL* (SVGF2SVGT.XSL);

- restituisce in uscita il documento SVG Tiny risultante.

Le regole elaborano il documento di origine trasferendo inalterate alcune componenti nel documento prodotto ed applicando, ove dovuto, le opportune trasformazioni. In particolare, le regole attuano, oltre alle trasformazioni sotto descritte, una *copia selettiva* degli elementi ed attributi presenti nel documento di origine: vengono copiati soltanto gli elementi e gli attributi conformi con la sintassi di SVG Tiny.

Le operazioni compiute nel processo di trasformazione sono le seguenti:

1. copia *selettiva* degli elementi e degli attributi contenuti nel documento SVG di origine:
   nel foglio di stile **SVGF2SVGT.xsl** sono definiti due insiemi in cui compaiono gli elementi e gli attributi previsti dalla sintassi SVG Tiny. Nel processo di copia gli elementi e gli attributi non appartenenti a tali insiemi sono scartati.
   È necessario però eseguire un ulteriore controllo: può verificarsi, infatti, che un certo attributo sia previsto nel profilo SVG Tiny *per un certo elemento*, ma non lo sia invece per un altro. Ciò accade, ad esempio, per l'attributo *font-stretch*, che non è ammesso nell'elemento *"text"*, ma lo è nell'elemento "*font-face*". Dall'analisi condotta risulta che un trattamento particolare deve essere riservato ai soli attributi "*font-stretch*" e "*font-variant*"; per la loro copia o eliminazione è necessario controllare a quale elemento appartengono: nel caso dell'elemento "*text*" sono scartati, nel caso di "*font-face*" copiati.
2. trasformazione dell'attributo "*style*":
   la descrizione dello stile di un elemento contenuta nella stringa dell'attributo "*style*" (sottoforma di una serie di coppie "*nome_attributo : valore_attributo*") è convertita in un insieme di attributi della forma "*nome_attributo* = *"valore_attributo*".
3. trasformazione relativa alla gestione dello stile di visualizzazione tramite CSS:
   la definizione dello stile, contenuta nell'elemento "*style*" e descritta tramite sintassi CSS, è recuperata ed inserita nell'elemento che appartiene alla classe corrispondente (selezionata tramite il valore dell'attributo "*class*", presente all'interno dell'elemento stesso).
4. casi particolari, attributo "*stroke-dasharray"*:
   se nel documento SVG di origine è presente un attributo "*stroke-dasharray*" con il valore "n,0", per evitare eventuali problemi di non visualizzazione, l'attributo non è copiato nel documento SVG Tiny di destinazione

Il processo di trasformazione è stato eseguito facendo uso del processore XSLT 1.0 contenuto nel package Microsoft XML Core Services (MSXML) 4.0.

5. Il servizio di download di documenti SVG Tiny

La produzione di documenti SVG Tiny ed il relativo download sono effettuati tramite il servizio web ***getmdmap*** (*get mobile device map*) in grado di fornire contenuti geografici relativi ad una mappa. Il servizio costituisce un'interfaccia verso il server Geomedia WebMap, accessibile dai dispositivi PDA e telefono cellulare. Esso viene invocato dal software residente sul dispositivo mobile tramite una richiesta di tipo HTTP GET[2] specificando la URL del servizio assieme ad alcuni di parametri:

http://webgisserver.isti.cnr.it/svgt/GetMPMap.asp?mapname=*mapname* & x=*x0* & y=*y0* &
dx=*width* & dy=*height* & compr={*true* | *false*}

Nella richiesta sono specificati, oltre al nome del servizio (*getmdmap.asp*), i seguenti parametri:

- *mapname*: nome della mappa che si vuole ottenere, definita nell'host come insieme di layer;
- *x0,y0* : coordinate del centro della finestra di mappa:
- *dx,dy* : offset dal centro della finestra di mappa sugli assi delle ascisse e delle ordinate (Figura 1);
- *compr*: assume valori "*true*|*false*"; è utilizzato per indicare se il servizio deve produrre un documento SVG oppure SVGZ.

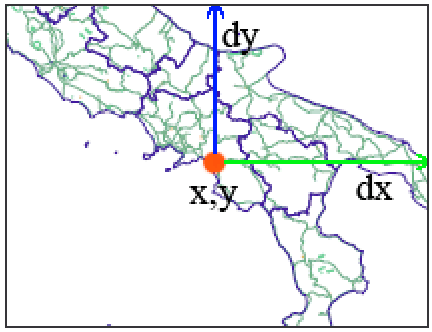


Figura 1 – Finestra di mappa e parametri identificativi

La rappresentazione di una mappa mediante SVG Tiny, da parte del servizio, avviene sfruttando il modello ad oggetti che GeoMedia WebMap offre; in base alla richiesta il servizio :

[2] Una richiesta di tipo http-GET si presenta nella forma http://host:[port]/path?{name=[value&]}, dove {} indica una lista di elementi e [] un elemento opzionale. Un parametro è rappresento secondo la codifica Key Value Pair (KVP) , quindi come *name=value* e separato dagli altri parametri dal carattere "&".

1. attiva una connessione al database contenente i dati geospaziali relativi alla mappa richiesta;
2. estrae la porzione di mappa relativa alla finestra di mappa specificata;
3. definisce la vestizione della mappa da restituire in base alle impostazioni definite sul server;
4. effettua la traduzione da primitive GIS a primitive SVG Full.

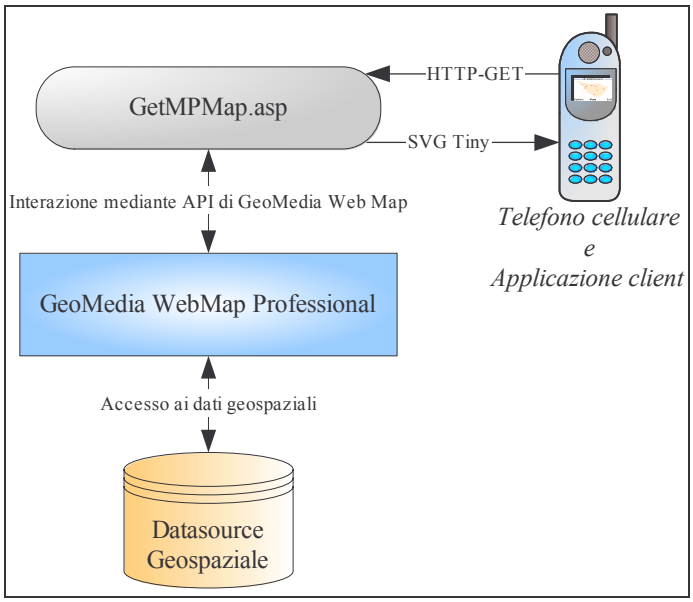


Figura 2 - Architettura del sistema, composto dal servizio GetMPMap e dispositivo cellulare, ed interazione fra sue le parti componenti.

Come detto in precedenza, la traduzione avviene in due passaggi; il secondo passaggio è di particolare rilievo in quanto permette di applicare qualsiasi trasformazione (descritta tramite il linguaggio XSLT) valida sul documento SVG. Nel nostro caso il servizio web utilizza il file *SVGF2SVGT.XSL* per far eseguire al server GeoMedia WebMap la trasformazione da SVG full a SVG Tiny. Una volta effettuata la trasformazione, viene prodotto un documento SVG Tiny in un file temporaneo che il servizio restituisce a chi ne a fatto richiesta; la pagina ASP ritorna un contenuto di tipo MIME *xml/svg* anziché *text/html*, quindi uno stream XML che il software che ne ha fatto richiesta può consumare. Nel caso vanga richiesto un file compresso (*svgz*), anziché uno stream XML, il servizio restituisce un documento con contenuto MIME *text/plain* contente la URL del file SVG Tiny.

Quanto detto è schematizzato dalla figura 2 in cui è riportata l'architettura del sistema, evidenziando l'interazioni fra le componenti.

6. La visualizzazione e l'integrazione sul dispositivo mobile

Per illustrare l'uso dell'applicazione residente sul dispositivo mobile, è stato utilizzato l'emulatore per telefoni cellulari J2ME Wireless Toolkit di Sun Microsystem; nella fattispecie è stato emulato il modello 6230 prodotto da Nokia in quanto supporta il profilo Mobile Information Device Platform (MIDP) 2.0 di J2ME (Java 2 Platform Micro Edition).
La mappa utilizzata per l'esempio, identificata sul server con il nome "toscana", è composta da tre strati informativi relativi alla regione Toscana rappresentanti rispettivamente le tre tipologie di feature geografiche (punto, linea,poligono):

- *citta*: rappresentai capoluoghi di provincia della regione;
- *fiumi* : i fiumi di maggiore importanza della regione;
- *regione* : limiti amministrativi.

L'applicazione client (simpleMapRequest) sul cellulare, per la composizione della richiesta HTTP-GET da inviare al servizio, richiede l'immissione della latitudine e longitudine, in decimal degree, relative al centro di mappa per simulare l'acquisizione della posizione effettuata da un generico sistema di localizzazione (figura 3). Il nome della mappa, le estensioni longitudinale e latitudinale dal centro di mappa (per contenere la mappa) ed il parametro di compressione sono definiti all'interno dell'applicazione .

Facendo riferimento alla richiesta mostrata nella paragrafo precedente, i parametri attuali specificati dall'applicazione sono:

- mapname: *toscana*, nome della mappa da visualizzare;
- x: *43.510* y: *11.065*, il centro di mappa cade all'interno della provincia di Firenze;
- dx: 1.335 dy: *1.121*, assieme alle coordinate di mappa permettono di definire un'area di interesse che contiene l'intera mappa;
- compr: *false*, il documento SVG Tiny trasferito non è compresso.

In base ai valori sopra descritti la richiesta si presenterà come segue:

http://webgisserver.isti.cnr.it/svgt/GetMPMap.asp? mapname=***toscana*** & x=***43.510*** & y=***11.065*** & dx=***1.335*** & dy=***1.121*** & compr=***false***

Figura 3 - Inserimento delle coordinate del centro di mappa mediante l'emulatore.

Una volta inviata la richiesta, il servizio "GetMPMap" restituisce un documento Tiny SVG contenente la porzione di mappa selezionata (figura 4). Sulla mappa visualizzata è possibile effettuare le seguenti operazioni di navigazione, accessibili mediante il menù "*Options*" (figure 3 e 5):

- *Pan*: mediante i tasti di navigazione del telefono cellulare è possibile spostare a finestra di visualizzazione sulla mappa;
- *Zoom*: operazioni di ingrandimento e riduzione dell'area visualizzata in modo da variarne il dettaglio (figura 6); l'operazione di ingrandimento (*zoom in*) permette di effettuare quattro livelli di ingrandimento;
- *Orig View*: permette di visualizzare l'estensione iniziale della mappa all'interno dell'area di visualizzazione del display.

Oltre a queste operazioni sulla mappa il menù possiede un comando per effettuare una nuova richiesta al servizio ("*Altra richiesta*") di download diun'altra mappa..

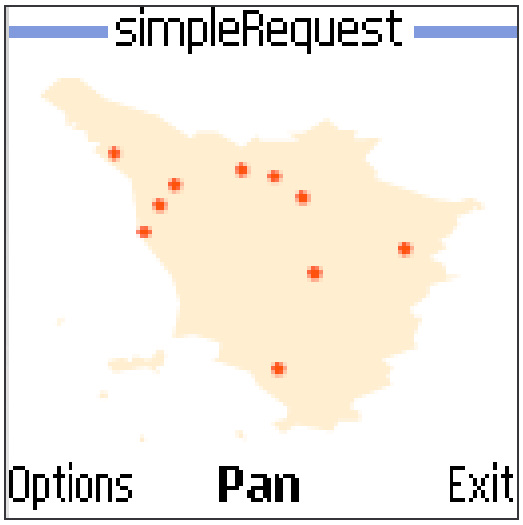


Figura 4 - Mappa della Toscana inviata dal servizio

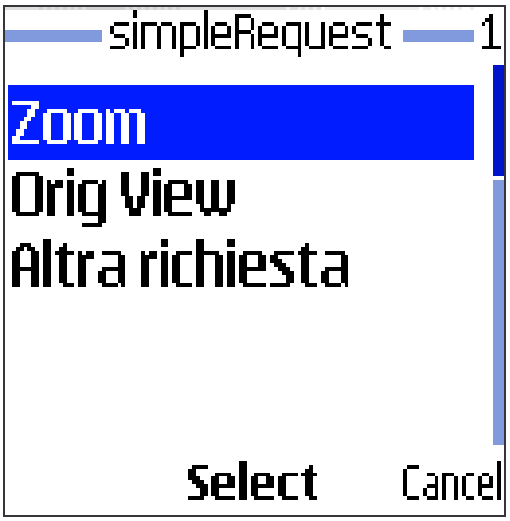


Figura 5- Menù "*Options*".

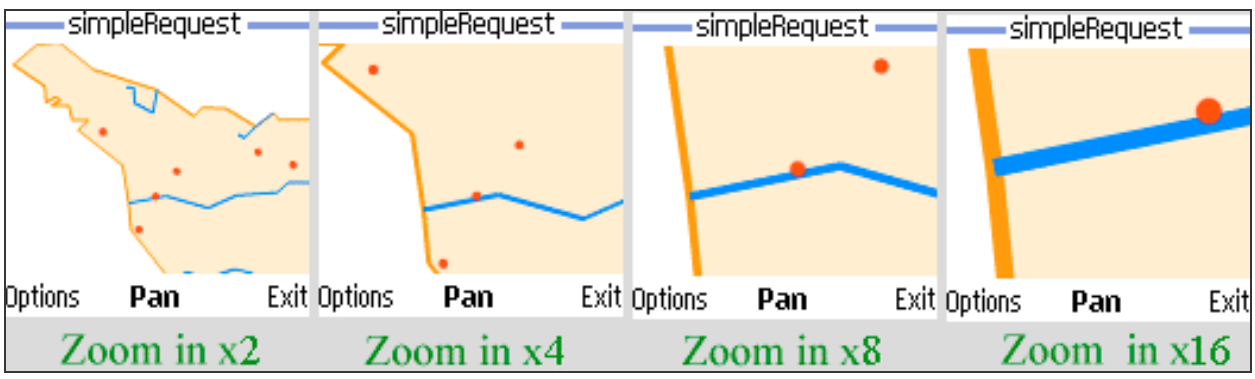


Figura 6 - Sequenza di quattro operazioni di ingrandimento, eseguite sulla mappa di figura 4.

Appendice A: Il file SVGF2SVGT.XSL

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:svg="http://www.w3.org/2000/svg" exclude-result-prefixes="svg">
   <xsl:output method="xml" indent="yes" encoding="UTF-8"/>
   <xsl:variable name="sezioneCD" select="//svg:style"/>
   <xsl:variable name="elementiTiny" select="'a, animate, animateColor, animateMotion,
      animateTransform, circle, defs, desc, ellipse, font-face, font-face-name, font-face-src,
      foreignObject, g, glyph, hkern, image, line, metadata, misssing-glyph, mpath, path, polygon,
      polyine, rect, set, svg, switch, symbol, text, title, use'"/>
   <xsl:variable name="attributiTiny" select="'accent-height, accumulate, additive, alphabetic,
      arabic-form, ascent, attributeName, attributeType, baseProfile, bbox, begin, by, calcMode,
      cap-height, color, color-rendering, content, cx, cy, d, descent, display, dur, end, fill,
      fill-rule, font-family, font-size, font-stretch, font-style, font-variant, font-weight,
      from, g1, g2, glyph-name, hanging, height, horiz-adv-x, horiz-origin-x, id, ideographic, k,
      keyPoints, keySplines, keyTimes, lang, mathematical, max, min, name, origin,
      overline-position, overline-thickness, panose-1, path, pathLength, points,
      preserveAspectRatio, r, repeatCount, repeatDur, requiredExtensions, requiredFeatures,
      restart, rotate, rx, ry, slope, stemh, stemv, strikethrough-position,
      strikethrough-thickness, stroke, stroke-dasharray, stroke-dashoffset, stroke-linecap,
      stroke-linejoin, stroke-miterlimit, stroke-width, systemLanguage, target, text-anchor, to,
      transform, type, u1, u2, underline-position, underline-thickness, unicode, unicode-range,
      units-per-em, values, version, viewBox, visibility, width, widths, x, x-height, x1, x2,
      xlink:actuate, xlink:arcrole, xlink:href, xlink:role, xlink:show, xlink:title, xlink:type,
      xml:base, xml:lang, xml:space, y, y1, y2, zoomAndPan'"/>
   <xsl:template match="/">
      <xsl:apply-templates/>
   </xsl:template>
   <xsl:template match="@*">
      <xsl:choose>
         <xsl:when test="local-name(current())='style'">
            <xsl:call-template name="attributiStyle">
               <xsl:with-param name="stringa" select="string(current())"/>
            </xsl:call-template>
         </xsl:when>
         <xsl:when test="local-name(current())='class'">
            <xsl:call-template name="selectClass">
               <xsl:with-param name="nomeClass" select="string(current())"/>
               <xsl:with-param name="nomeElem" select="local-name(parent::*)"/>
            </xsl:call-template>
         </xsl:when>
         <xsl:when test="local-name(current())='stroke-dasharray'">
            <xsl:variable name="dopoVirgola"
               select="normalize-space(substring-after(string(current()),','))"/>
            <xsl:if test="$dopoVirgola!='0'">
               <xsl:copy/>
            </xsl:if>
         </xsl:when>
         <xsl:otherwise>
            <xsl:if test="contains($attributiTiny, name(current()))">
               <xsl:copy/>
            </xsl:if>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:apply-templates select="@*|node()"/>
   </xsl:template>
   <xsl:template match="node()">
      <xsl:choose>
         <xsl:when test="local-name(current())='style'"/>
         <xsl:when test="contains($elementiTiny, name(current()))">
            <xsl:copy>
               <xsl:call-template name="selectElemClass">
                  <xsl:with-param name="nomeElem"
                     select="normalize-space(local-name(current()))"/>
               </xsl:call-template>
               <xsl:apply-templates select="@*|node()"/>
            </xsl:copy>
         </xsl:when>
         <xsl:otherwise/>
      </xsl:choose>
   </xsl:template>
   <xsl:template match="svg:svg">
      <xsl:copy>
         <xsl:apply-templates select="@*"/>
         <xsl:apply-templates select="node()"/>
      </xsl:copy>
   </xsl:template>
   <xsl:template name="attributiStyle">
      <xsl:param name="stringa"/>
      <xsl:choose>
         <xsl:when test="contains($stringa,';')">
            <xsl:variable name="primaCoppia" select="substring-before($stringa,';')"/>
            <xsl:variable name="successive" select="substring-after($stringa,';')"/>
            <xsl:variable name="nomeAttributo"
               select="normalize-space(substring-before($primaCoppia,':'))"/>
            <xsl:if test="contains($attributiTiny, $nomeAttributo)">
               <xsl:choose>
                  <xsl:when test="$nomeAttributo='stroke-dasharray'">
                     <xsl:call-template name="doStrokeDash">
                        <xsl:with-param name="valoreStrokeDash"
```

```
                        select="normalize-space(substring-after($primaCoppia,':'))"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="{$nomeAttributo}">
                    <xsl:value-of select="substring-after($primaCoppia, ':')"/>
                </xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
    <xsl:call-template name="attributiStyle">
        <xsl:with-param name="stringa" select="$successive"/>
    </xsl:call-template>
</xsl:when>
<xsl:otherwise>
    <xsl:variable name="nomeAttributo"
        select="normalize-space(substring-before($stringa,':'))"/>
    <xsl:if test="contains($attributiTiny, $nomeAttributo)">
        <xsl:choose>
            <xsl:when test="$nomeAttributo='stroke-dasharray'">
                <xsl:call-template name="doStrokeDash">
                    <xsl:with-param name="valoreStrokeDash"
                        select="normalize-space(substring-after($stringa,':'))"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="{$nomeAttributo}">
                    <xsl:value-of select="substring-after($stringa, ':')"/>
                </xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
<xsl:template name="selectClass">
    <xsl:param name="nomeClass"/>
    <xsl:param name="nomeElem"/>
    <xsl:variable name="nomeCompletoClass" select="concat($nomeElem, '.', $nomeClass)"/>
    <xsl:variable name="maybeCoppie" select="substring-after($sezioneCD, $nomeCompletoClass)"/>
    <xsl:choose>
        <xsl:when test="string-length($maybeCoppie)!=0">
            <xsl:variable name="coppieConGraffa" select="substring-before($maybeCoppie,'}')"/>
            <xsl:call-template name="creaAttributiClass">
                <xsl:with-param name="stringaAtt" select="substring-after($coppieConGraffa,'{')"
                />
            </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
            <xsl:variable name="coppieAfter" select="substring-after($sezioneCD, $nomeClass)"/>
            <xsl:variable name="coppieConGraffa" select="substring-before($coppieAfter,'}')"/>
            <xsl:call-template name="creaAttributiClass">
                <xsl:with-param name="stringaAtt" select="substring-after($coppieConGraffa,'{')"
                />
            </xsl:call-template>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
<xsl:template name="selectElemClass">
    <xsl:param name="nomeElem"/>
    <xsl:variable name="maybeCoppie"
        select="normalize-space(substring-after($sezioneCD,$nomeElem))"/>
    <xsl:if test="starts-with($maybeCoppie, '{')">
        <xsl:variable name="coppieConGraffa" select="substring-before($maybeCoppie,'}')"/>
        <xsl:call-template name="creaAttributiClass">
            <xsl:with-param name="stringaAtt" select="substring-after($coppieConGraffa,'{')"/>
        </xsl:call-template>
    </xsl:if>
</xsl:template>
<xsl:template name="creaAttributiClass">
    <xsl:param name="stringaAtt"/>
    <xsl:variable name="primaCoppia" select="substring-before($stringaAtt,';')"/>
    <xsl:variable name="nomeAttributo"
        select="normalize-space(substring-before($primaCoppia,':'))"/>
    <xsl:if test="contains($attributiTiny, $nomeAttributo)">
        <xsl:choose>
            <xsl:when test="normalize-space(string($nomeAttributo))='stroke-dasharray'">
                <xsl:call-template name="doStrokeDash">
                    <xsl:with-param name="valoreStrokeDash"
                        select="normalize-space(substring-after($primaCoppia,':'))"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="{$nomeAttributo}">
                    <xsl:value-of select="substring-after($primaCoppia,':')"/>
                </xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
    <xsl:if test="contains(substring-after($stringaAtt,';'),':')">
        <xsl:variable name="successive" select="substring-after($stringaAtt,';')"/>
```

```
            <xsl:call-template name="creaAttributiClass">
               <xsl:with-param name="stringaAtt" select="$successive"/>
            </xsl:call-template>
         </xsl:if>
      </xsl:template>
      <xsl:template name="doStrokeDash">
         <xsl:param name="valoreStrokeDash"/>
         <xsl:variable name="dopoVirgola"
            select="normalize-space(substring-after($valoreStrokeDash,','))"/>
         <xsl:if test="$dopoVirgola!=0">
            <xsl:attribute name="stroke-dasharray">
               <xsl:value-of select="$valoreStrokeDash"/>
            </xsl:attribute>
         </xsl:if>
      </xsl:template>
   </xsl:stylesheet>
```

Appendice B: Il file SVG Full

```
<?xml version="1.0"  encoding="UTF-8"?>
<svg id="gwmroot" preserveAspectRatio="xMidYMid meet" viewBox="0 0 4800 6400"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gmwmsvg="http://www.intergraph.com/GeoMedia/wmsvg" style="stroke-linejoin:round">
<mask id="gwmmask">
<rect id="gwmmaskrect" x="0" y="0" width="4800" height="6400" style="fill:white"/>
</mask>
<g id="_GWMAll" gmwmsvg:typ="a" mask="url(#gwmmask)">
<metadata><gmwmsvg:metadata id="_GWMMetadata" gmwmsvg:storageoffsetx="9.698007"
 gmwmsvg:storageoffsety="45.139049"
 gmwmsvg:storagetoreadoutscale="1.000000"
 gmwmsvg:readoutoffsetx="0.000000"
 gmwmsvg:readoutoffsety="0.000000"
 gmwmsvg:readoutunit="m"
 gmwmsvg:storagetodistancescale="1.000000"
 gmwmsvg:distanceunit="m"
 gmwmsvg:displaytostoragescale="1797.058216"
 gmwmsvg:version="1.0"/></metadata><rect id="gwmbg" x="-48000" y="-64000" width="100800" height="134400"
style="fill:#ffffff"/>
<g id="BGROUND
http://;1.000000;9.698007;45.139049;0.000556;1.000000;0.000000;1.000000;0.000000;m;1.000000;m"
gmwmsvg:typ="l"/>
<g id="regione" gmwmsvg:typ="l" gmwmsvg:pri="100" class="i7t3j0k0T4" gmwmsvg:hi="i7t3j0k0T4 t9K4w2x4l9">
<path d="M0 1382l 218-181 199-16 305 243 569 180 2 3 329 229 78-75 211 8 270 159 156-120 52 65 418-7-133-88
168-6 253-148 143 129 166 1-17 90 295-20-191 227 136 134-14 78 615 215 259 44 160-59 153 96-98 58-24-85-21
79-182 70 39 56-103 143-23-6-93 61 93 55-203 112 166 81 23 123 165 3-352 100-4 88-195 151 36 92 89 8-96 222
45 96-375 125 112 97-56 75 50 101-411 156-37 90 107 50-6 93-223-6-69 110-394-68-143 93-117-120 163-42 10-
120-316-234-80-135-420-133 76-40 15-141-293-89-216 39 68-374-88-285-360-457-24-237-49-340-400-313 92-89-149
3-23-129-161-2 81-50-46-24-86 44-11-143-283-193zM733 4245l 31-39 134-25 86 47 36-74 159 59 117-131 58 47-16
122-91 30 63 89-151-98-358 68-68-95zM2103 4992l 5-21 26 13-1-42 48 19 40 95-118-64zM180 3803l 85-71-35 104-
50-33zM632 4593l 56-65 24 7 26 48-25 38-81-28zM1076 5044l 3-21 29-10 35 43-53 27-14-39zM2515 5183l 35-5-8
37-30-26 3-6z" id="1"/>
</g>
<g id="fiumi" gmwmsvg:typ="l" gmwmsvg:pri="100" class="X9Q7c4v0o6" gmwmsvg:hi="X9Q7c4v0o6 o4t3a0s8m5">
<path d="M1035 2618l 338-68 380 107 250-110 365-25 99-87 578-9 125 54-23 202 181 173 455 82" id="4"/>
<path d="M1419 3312l 141-84 173 58 251-83 145 33 216 123 33 172-68 75" id="10"/>
<path d="M3194 3600l-193 110 34 176-202 87 75 149-551 348" id="9"/>
<path d="M3783 2937l 55 507 150 208 36 92" id="6"/>
<path d="M3783 2937l 107-36 35-129-405-416 49-60" id="5"/>
<path d="M3239 3134l 148 170-193 296" id="8"/>
<path d="M4228 2482l-5 318 165 78 23 6" id="7"/>
<path d="M2337 1812l-156 120-105 105-58-37" id="1"/>
<path d="M1291 1608l 2 3-38 198-164-85-7-133-68-10" id="2"/>
<path d="M3404 1758l-209 170" id="3"/>
</g>
<g id="citta" gmwmsvg:typ="l" gmwmsvg:pri="100" class="q0T6w2c0I5" gmwmsvg:hi="q0T6w2c0I5 c1y5n1U1m9">
<rect x="2937" y="3268" width="20" height="20" rx="5" ry="5" id="1"/>
<rect x="3917" y="3000" width="20" height="20" rx="5" ry="5" id="2"/>
<rect x="798" y="1982" width="20" height="20" rx="5" ry="5" id="3"/>
<rect x="1453" y="2315" width="20" height="20" rx="5" ry="5" id="4"/>
<rect x="1265" y="2544" width="20" height="20" rx="5" ry="5" id="5"/>
<rect x="2178" y="2151" width="20" height="20" rx="5" ry="5" id="6"/>
<rect x="2505" y="2250" width="20" height="20" rx="5" ry="5" id="7"/>
<rect x="2789" y="2431" width="20" height="20" rx="5" ry="5" id="8"/>
<rect x="1111" y="2842" width="20" height="20" rx="5" ry="5" id="9"/>
<rect x="2544" y="4263" width="20" height="20" rx="5" ry="5" id="10"/>
</g>
<style type="text/css" id="gwmstyles"><![CDATA[
.i7t3j0k0T4
{
stroke:#466271;
fill:#1ffe16;
fill-opacity:1;
stroke-width:20;
stroke-dasharray:1,0;
pointer-events:visible;
}
.t9K4w2x4l9
{
stroke:#ff0000;
fill:#ff0000;
fill-opacity:1;
stroke-width:20;
stroke-dasharray:1,0;
pointer-events:visible;
}
.X9Q7c4v0o6
{
stroke:#9bc7f2;
fill:none;
stroke-width:20;
stroke-dasharray:1,0;
pointer-events:visiblePainted;
}
.o4t3a0s8m5
{
stroke:#ff0000;
```

```
fill:none;
stroke-width:20;
stroke-dasharray:1,0;
pointer-events:visiblePainted;
}
.q0T6w2c0I5
{stroke:none;
fill:#5d66d7;
}
.c1y5n1U1m9
{stroke:none;
fill:#ff0000;
}
]]></style>
</g>
</svg>
```

Appendice C: Il file SVG Tiny

```
<?xml version="1.0" encoding="UTF-8"?>
<svg id="gwmroot" preserveAspectRatio="xMidYMid meet" viewBox="0 0 4800 6400" stroke-linejoin="round"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gmwmsvg="http://www.intergraph.com/GeoMedia/wmsvg">
    <g id="_GWMAll">
        <metadata></metadata>
        <rect id="gwmbg" x="-48000" y="-64000" width="100800" height="134400" fill="#ffffff"></rect>
        <g id="BGROUND
http://;1.000000;9.698007;45.139049;0.000556;1.000000;0.000000;1.000000;0.000000;m;1.000000;m"></g>
        <g id="regione" stroke="#466271" fill="#1ffe16" stroke-width="20">
            <path d="M0 1382l 218-181 199-16 305 243 569 180 2 3 329 229 78-75 211 8 270 159 156-120 52 65
418-7-133-88 168-6 253-148 143 129 166 1-17 90 295-20-191 227 136 134-14 78 615 215 259 44 160-59 153 96-98
58-24-85-21 79-182 70 39 56-103 143-23-6-93 61 93 55-203 112 166 81 23 123 165 3-352 100-4 88-195 151 36 92
89 8-96 222 45 96-375 125 112 97-56 75 50 101-411 156-37 90 107 50-6 93-223-6-69 110-394-68-143 93-117-120
163-42 10-120-316-234-80-135-420-133 76-40 15-141-293-89-216 39 68-374-88-285-360-457-24-237-49-340-400-313
92-89-149 3-23-129-161-2 81-50-46-24-86 44-11-143-283-193zM733 4245l 31-39 134-25 86 47 36-74 159 59 117-
131 58 47-16 122-91 30 63 89-151-98-358 68-68-95zM2103 4992l 5-21 26 13-1-42 48 19 40 95-118-64zM180 3803l
85-71-35 104-50-33zM632 4593l 56-65 24 7 26 48-25 38-81-28zM1076 5044l 3-21 29-10 35 43-53 27-14-39zM2515
5183l 35-5-8 37-30-26 3-6z" id="1"></path>
        </g>
        <g id="fiumi" stroke="#9bc7f2" fill="none" stroke-width="20">
            <path d="M1035 2618l 338-68 380 107 250-110 365-25 99-87 578-9 125 54-23 202 181 173 455 82"
id="4"></path>
            <path d="M1419 3312l 141-84 173 58 251-83 145 33 216 123 33 172-68 75" id="10"></path>
            <path d="M3194 3600l-193 110 34 176-202 87 75 149-551 348" id="9"></path>
            <path d="M3783 2937l 55 507 150 208 36 92" id="6"></path>
            <path d="M3783 2937l 107-36 35-129-405-416 49-60" id="5"></path>
            <path d="M3239 3134l 148 170-193 296" id="8"></path>
            <path d="M4228 2482l-5 318 165 78 23 6" id="7"></path>
            <path d="M2337 1812l-156 120-105 105-58-37" id="1"></path>
            <path d="M1291 1608l 2 3-38 198-164-85-7-133-68-10" id="2"></path>
            <path d="M3404 1758l-209 170" id="3"></path>
        </g>
        <g id="citta" stroke="none" fill="#5d66d7">
            <rect x="2937" y="3268" width="20" height="20" rx="5" ry="5" id="1"></rect>
            <rect x="3917" y="3000" width="20" height="20" rx="5" ry="5" id="2"></rect>
            <rect x="798" y="1982" width="20" height="20" rx="5" ry="5" id="3"></rect>
            <rect x="1453" y="2315" width="20" height="20" rx="5" ry="5" id="4"></rect>
            <rect x="1265" y="2544" width="20" height="20" rx="5" ry="5" id="5"></rect>
            <rect x="2178" y="2151" width="20" height="20" rx="5" ry="5" id="6"></rect>
            <rect x="2505" y="2250" width="20" height="20" rx="5" ry="5" id="7"></rect>
            <rect x="2789" y="2431" width="20" height="20" rx="5" ry="5" id="8"></rect>
            <rect x="1111" y="2842" width="20" height="20" rx="5" ry="5" id="9"></rect>
            <rect x="2544" y="4263" width="20" height="20" rx="5" ry="5" id="10"></rect>
        </g>
    </g>
</svg>
```

Riferimenti

- W3C Scalable Vector Graphics (SVG) 1.1 Specification
  http://www.w3.org/TR/SVG/

- W3C Scalable Vector Graphics (SVG) Full 1.2 Specification
  http://www.w3.org/TR/2005/WD-SVG12-20050413/

- W3C Mobile SVG Profiles: SVG Tiny and SVG Basic
  http://www.w3.org/TR/SVGMobile/