Capítulo 3

Conversión de SVG a SWF

3.1 Cómo convertir de SVG a SWF

Hemos establecido ya toda la teoría necesaria para llevar a cabo una transformación de SVG a SWF. Ahora veremos cómo es que se puede convertir de un formato a otro.

En el capítulo 2, *Marco Teórico*, mencionamos tres tipos de formatos diferentes: SVG, SWF y FXML. Utilizaremos el formato FXML como nuestro intermediario entre SVG y SWF, por lo que necesitaremos realizar dos conversiones:

La primera trata de convertir el formato SVG al formato FXML y la segunda trata de convertir el formato FXML a SWF. El archivo FXML puede ser el resultado de combinar varios archivos SVG y acomodarlos en “capas” en el archivo SWF. Debido a que se necesitaría más tiempo del que se tomó para realizar este proyecto de tesis, no se incluyeron variaciones como interactividad con el usuario para activar y desactivar capas del mapa, o que al señalar con el *Mouse* algún vector, nos indique el nombre de la zona. Todos estos efectos podrían ser muy interesantes y fáciles de acoplar al proyecto como trabajos futuros sobre el mismo.

En adelante se llamará SVG-FXML al módulo que realiza la conversión entre estos dos formatos y FXML-SWF al que realiza la segunda conversión.

Es importante destacar que no serán cubiertas por completo las especificaciones de los dos formatos (SVG y Flash) , ya que ambos son muy extensos; por lo que se tratará de

abarcar los subconjuntos necesarios para representar la mayoría de los documentos cartográficos.

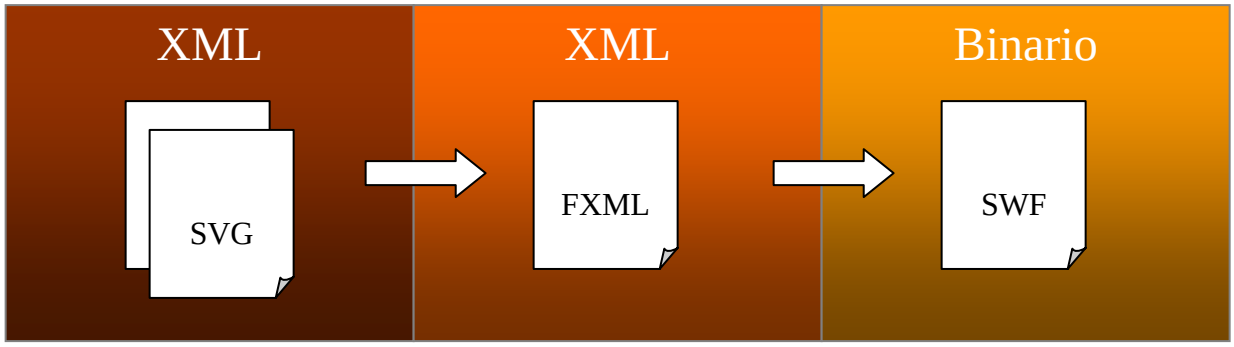

Figura 3.1 La conversión pasa por FXML como archivo intermedio

3.1.2 Arquitectura del sistema

A fin de ofrecer el servicio de conversión, el programa estará disponible en una aplicación Web. Dicha aplicación permitirá el acceso de los diferentes dispositivos con los que se pretende hacer las pruebas, dado que todos ellos cuentan con un navegador Web.

Debido a que la intención de la aplicación Web es solamente realizar pruebas sobre la conversión de los documentos, la fuente de los archivos SVG para convertir será obtenida tal cual del sistema de archivos del servidor donde resida la aplicación, no se obtienen de una fuente de datos estructurada como una *base de datos*; ese tipo de trabajo ya ha sido realizado por otros autores de tesis, como Espinosa [Espinosa-2002] y Rodríguez [Rodríguez-2004]. No se utilizaron los proyectos de su creación puesto que la elección de

la tecnología para este proyecto es diferente a la que ellos utilizaron en su momento y no podrían operar entre sí.

La arquitectura está definida entonces, de la siguiente manera: el cliente accederá al sistema por Web para solicitar uno o más documentos ya convertidos puesto que el proceso de conversión puede ser muy tardado y serán enviados al cliente, el cual los podrá visualizar con el reproductor de Flash, el cual guardará los mapas de manera *temporal*. Se ofrecerán algunas opciones avanzadas sobre la personalización de la conversión.

El navegador en el dispositivo cliente guardará la copia temporal hasta que el *caché* de navegación sea limpiado, y como se comentó en el segundo capítulo contará con las funciones básicas del visualizador: aumentar, reducir, explorar (arrastrar y soltar) y elegir la calidad de trazo entre baja, mediana y alta.

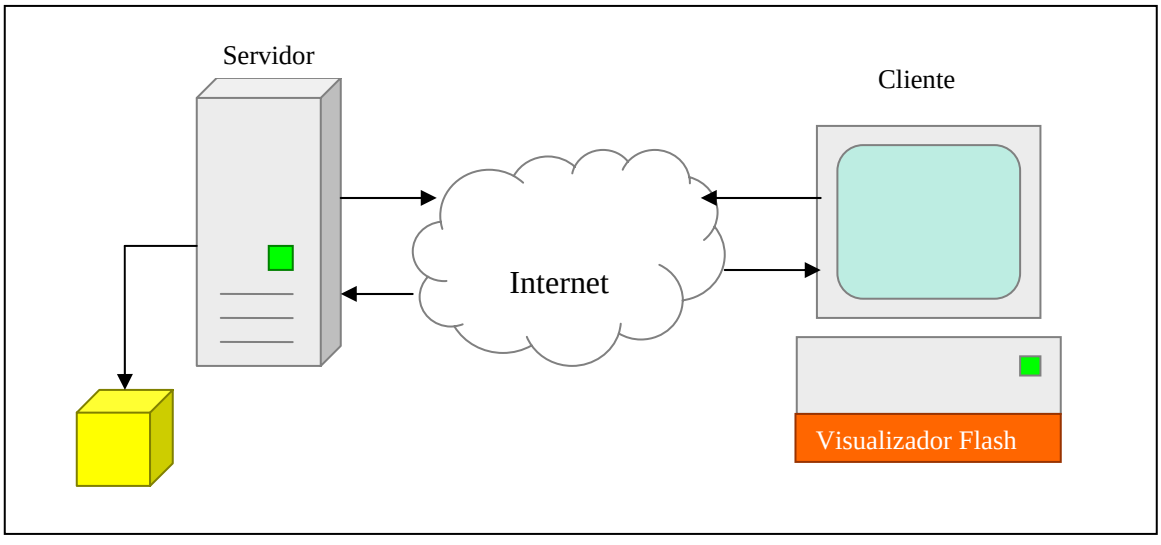

Figura 3.2 Arquitectura del sistema.

3.1.3 Tecnología a utilizar

Se ha elegido a la plataforma .NET de Microsoft como tecnología, tanto para el motor de conversión, como para la aplicación basada en Web, ya que el entorno de programación que Microsoft ofrece para dicha plataforma (Visual Studio) resulta muy cómodo y funcional para el programador.

Entre las opciones a utilizar, se tuvo a PHP como tecnología para el desarrollo del proyecto, debido a que éste cuenta [PHPFlash-2005] con una librería de producción de archivos con el formato SWF. Esta opción acabó descartada debido a que el autor de la librería SWF ha abandonado el desarrollo de la misma, la cual se encuentra incompleta y en estado experimental. Esto fue lo que motivó el estudio profundo de la especificación del formato de SWF.

Específicamente, se utilizó el lenguaje Visual Basic .NET (VB.NET) para el desarrollo del convertidor y ASP.NET para la aplicación Web. La interpretación de XML está a cargo de la librería XML.NET de la plataforma. La lectura del formato binario de SWF será byte por byte con las herramientas nativas del lenguaje VB.NET

3.2 Convertir de SVG a FXML

3.2.1 Los encabezados de los formatos

Los encabezados de ambos formatos como tal, definen que se trata de un documento de XML, por lo que tomaremos como encabezado del documento FXML al contenido de la etiqueta *Header*, la cual debe de ser la primera en aparecer dentro del nodo raíz (*swf*); y como encabezado del documento SVG a la etiqueta raíz (*svg*).

Entonces pues, el encabezado de FXML necesita la siguiente información:

- **Firma del formato**. El módulo FXML-SWF sólo admite la firma FWS, ya que la firma CWS indica que la información se encuentra comprimida.
- **Tamaño del lienzo**. Más abajo se discute sobre la interpretación del mismo.
- **Versión**. El módulo FXML-SWF está diseñado para escribir archivos SWF de versión 6 o 7.
- **Total de cuadros**. Debido a que los documentos que se transformarán no contienen animación, este campo estará dado siempre por un cuadro.
- **Cuadros por segundo**. Será establecido a 0.12 (12 cuadros por segundo) por costumbre en los archivos SWF. Podría ser cualquier valor numérico.

En el formato SVG, es opcional especificar el tamaño del lienzo. Los autores de SVG dejan el tamaño predeterminado del lienzo a decisión de los autores de los

visualizadores. También se puede especificar el atributo vista (*viewbox)*, el cual es un rectángulo que permite delimitar la zona de trabajo. Esta vista también puede servir para alejar o acercar la escena por primera instancia.

Esto es, que si se define una vista de (100, 100) a (2000, 2000) y un lienzo de 300 x 300 píxeles, al dibujar una línea que va de (100, 100) a (2000, 2000), el visualizador deberá pintar la línea en el lienzo de (0,0) a (300, 300).

En el formato FXML el tamaño del lienzo se declara en un rectángulo también, pero a diferencia de SVG, este rectángulo debe de tener su ubicación forzosamente en (0,0) y deberá de definir el tamaño del lienzo en la segunda coordenada.

Debido a todo esto, el módulo SVG-XML establece un tamaño de lienzo predeterminado de 800 píxeles de ancho por 600 píxeles de alto. Este módulo también debe de recordar si se especificó una *vista*, ya que cada coordenada será transportada al lienzo en FXML de la siguiente manera:

X’ = (X – Vista.X) * Lienzo.Ancho / Vista.Ancho

Y’ = (Y – Vista.Y) * Lienzo.Alto / Vista.Alto

Así, todos los puntos especificados en el espacio SVG, son transportados al espacio FXML. La desventaja de esta operación es la posible pérdida de la proporción ancho-alto si se alteran de manera errónea las dimensiones del lienzo FXML.

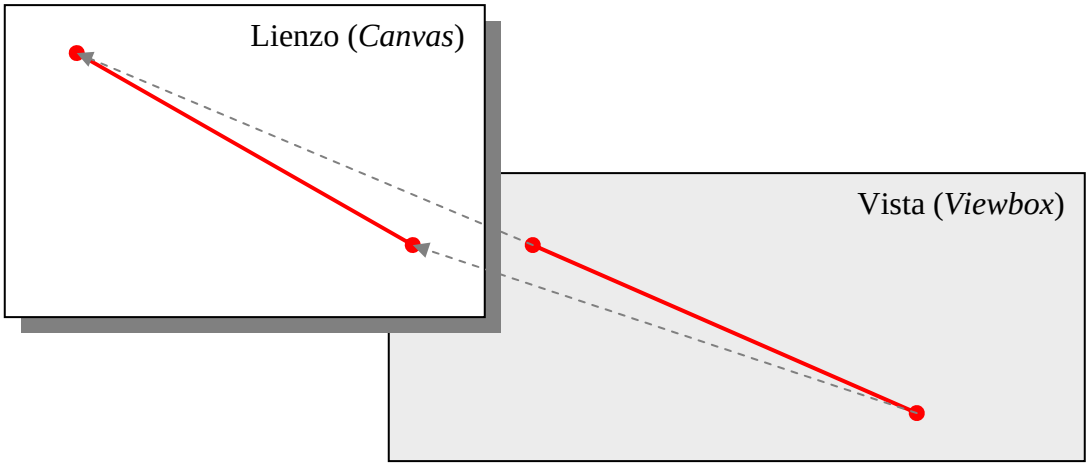

Figura 3.3 La vista debe de ser proyectada sobre el lienzo

3.2.2 Transformaciones Geométricas

En el segundo capítulo de este documento, observamos cómo SVG puede anidar transformaciones en los puntos mediante la especificación del atributo transformar (*transform*) en los elementos que lo permiten, estableciendo una matriz de transformación, o transformaciones específicas como trasladar, girar o escalar.

El formato FXML también permite especificar una matriz de transformación mediante el elemento MATRIX (Ver apéndice A). No obstante, el módulo SVG-FXML realiza las transformaciones especificadas en el documento con dos fines:

- **Generalizar el posicionamiento** de los puntos, lo cual se hace también con la *vista* especificada

- **Evitar procesamiento extra** en el cliente. Debido a que se tiene contemplada la visualización de los resultados en dispositivos móviles, evitaremos a toda costa delegar procesamiento extra al cliente. Esta es una filosofía que se guardó durante todo el proyecto.

Esto se logra mediante una estructura de pila. Esta estructura almacena la CTM (Matriz de transformación actual) hasta el punto en que cada una de las etiquetas con un atributo *transformar* aparece. De esta manera, la estructura en la punta de la pila, siempre contendrá la CTM correcta. En la estructura de la CTM se almacena también el nombre de la etiqueta que contenía al atributo y la *profundidad* de la misma, así, al encontrar la etiqueta de cierre con el mismo nombre y profundidad, se realiza una operación *pop* a la pila y la estructura en la punta ahora contiene la CTM adecuada. Esto es ilustrado en la siguiente figura

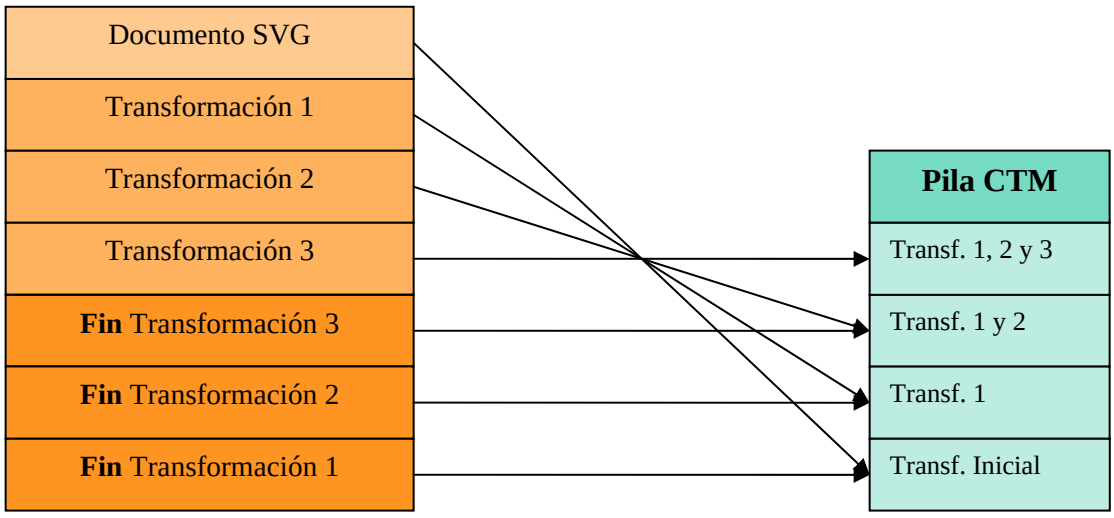

Figura 3.4 Una estructura de pila conserva la CTM indicada.

Así la pila de CTM contiene para cada nivel, la concatenación de las matrices de transformación anteriores. La concatenación de las matrices consiste en multiplicar la CTM por la matriz de transformación especificada. Para más detalle en las matrices de transformación consulte el capítulo 2, sección El formato SVG, Transformaciones.

3.2.3 Conversión de Líneas y Polilíneas

A fin de generalizar las figuras en el formato, una etiqueta *DefineShape* y otra *PlaceObject2* es generada por cada uno de los elementos encontrados del tipo: ***line,*** ***polyline*** y ***path.***

En el contexto de los documentos cartográficos, es muy recomendable utilizar la etiqueta *polyline* (poli-línea), pues así obtendremos trazos más largos y por lo tanto, archivos SWF más pequeños.

Esta sugerencia se debe al hecho de que cada etiqueta *DefineShape* trae consigo la elección del estilo de línea y de relleno, así como su definición; y la etiqueta *PlaceObject2* ocupa cierto espacio también, muy considerable al insertarla miles de veces.

Es por todo lo anterior que no es muy recomendable utilizar la etiqueta *line* para describir una línea, cuando lo que se busca es seguir un trazo compuesto de varias líneas, pues esto generaría mucha basura en el archivo FXML y por lo tanto en el archivo SWF.

Dado que este proyecto es limitado en la especificación, el atributo d (datos) en la etiqueta *path* (trazo), se ignoran los comandos M, L y Z (Mover a, línea a, y cerrar figura). El módulo SVG-FXML asumirá que el primer par de coordenadas es el comando *mover a*, y en adelante las coordenadas serán tomadas como *líneas a*.

Las coordenadas del trazo en *polyline* o en *path*, son establecidas de manera diferente a las que se deben de especificar en el formato FXML, ya que en este último es necesario especificar, primero que nada, hacia donde mover la pluma, y consecuentemente los incrementos o decrementos en las coordenadas del trazo. Debido a esto tenemos que colocar la diferencia de los puntos SVG en los trazos del formato FXML. Estas diferencias son mencionadas en la especificación como *DeltaX* y *DeltaY*.

Tabla de traducción (líneas)

Elemento en SVG	Elemento en FXML
<line x1=″<x1>″ y1=″<y1>″ x2=″<x2>″ y2=″<y2>″ />	<DefineShape> <ShapeId>1</ShapeId> <RECT> <XMin>x1</XMin> <XMax>x2</XMax> <YMin>y1</YMin> <YMax>y2</YMax> </RECT> <SHAPEWITHSTYLE> <FILLSTYLEARRAY> <FillStyleCount>0</FillStyleCount> </FILLSTYLEARRAY> <LINESTYLEARRAY> <LineStyleCount>1</LineStyleCount> <LINESTYLE> <Width>0.1</Width> <RGB> <R>0</R> <G>0</G> <B>0</B> </RGB> </LINESTYLE> </LINESTYLEARRAY> <STYLECHANGERECORD> <StateNewStyles>0</StateNewStyles> <StateLineStyle>1</StateLineStyle> <StateFillStyle1>0</StateFillStyle1> <StateFillStyle0>0</StateFillStyle0> <StateMoveTo>1</StateMoveTo> <MoveDeltaX>x1</MoveDeltaX> <MoveDeltaY>y1</MoveDeltaY> <LineStyle>1</LineStyle> </STYLECHANGERECORD> <STRAIGHTEDGERECORD> <LineType>General</LineType>

	<DeltaX>x2</DeltaX> <DeltaY>y2</DeltaY> </STRAIGHTEDGERECORD> <ENDSHAPERECORD /> </SHAPEWITHSTYLE> </DefineShape> <PlaceObject2> <PlaceFlagHasClipActions>0</PlaceFlagHasClipActions> <PlaceFlagHasClipDepth>0</PlaceFlagHasClipDepth> <PlaceFlagHasName>0</PlaceFlagHasName> <PlaceFlagHasRatio>0</PlaceFlagHasRatio> <PlaceFlagHasColorTransform>0</PlaceFlagHasColorTransform> <PlaceFlagHasMatrix>1</PlaceFlagHasMatrix> <PlaceFlagHasCharacter>1</PlaceFlagHasCharacter> <PlaceFlagMove>0</PlaceFlagMove> <Depth>1</Depth> <CharacterId>1</CharacterId> <MATRIX> <EmptyMatrix /> </MATRIX> </PlaceObject2>
<polyline points="{lista de puntos}"> <path d="{comandos de líneas}">	Lo mismo que para line, con un EDGERECORD por cada punto en el polígono

Podemos observar como FXML resulta en archivos más grandes que los originales en SVG. Esto se debe a que FXML tiene la instrucción de pintado de la línea que será convertido a bits. El archivo FXML puede ser destruido tras finalizar la conversión entre los formatos.

3.2.4 Conversión de estilo (Formato rico de geometría)

Al convertir documentos cartográficos a SVG, usualmente se hace uso de la etiqueta de agrupamiento *g*, la cual engloba el estilo de la capa en cuestión. Debido al comportamiento que facilita XML, la anidación, este estilo también debe de ser acumulado durante las diferentes capas del documento y es mediante una *pila*, muy similarmente a la pila que se utiliza para acumular información sobre la CTM (transformación). Por supuesto este programa no soporta todos los comandos de estilo. Los comandos soportados son:

Comando	**Uso**

Stroke-color	Color del pincel
Stroke-width	Ancho del pincel
Fill-color	Color de relleno

3.3 Convertir de FXML a SWF

Convertir FXML a SWF es muy sencillo, ya que por su estructura FXML es la clara instrucción de qué información plasmar en bits en el formato SWF (Ver capítulo 2).

3.3.1 Conversiones de unidades de medida

Con fines de optimizar el trazado, SWF utiliza todas las coordenadas en *twips*, que son una medida más pequeña que el píxel. Un píxel equivale a 20 twips. Para facilitar la interpretación humana de FXML, éste requiere que las coordenadas se encuentren en píxeles.

Debido a esto, cada coordenada obtenida de FXML es multiplicada por 20 para obtener su valor en *twips.* Si el valor en *twips* es de punto flotante, será redondeado.

3.3.2 Trabajo a futuro

En el desarrollo de este documento se han mencionado las capacidades del formato SWF. Con el tiempo, este formato ha crecido a diferentes áreas informáticas, en gran parte por medio del *Scripting* que soporta *Flash Player*. Debido a que este trabajo solamente introdujo el tema del uso de SWF en temas cartográficos para la fácil transmisión y visualización de los mismos. Este documento intenta ser una guía para cualquier autor que

desee continuar o completar el trabajo realizado a fin de lograr metas más allá de las fronteras de la simple visualización o más allá de los documentos cartográficos.

Este documento propone posibles trabajos a futuro en dos grandes áreas:

- **Documentos Cartográficos.** Al completar la lectura/escritura en SWF y la estructura correspondiente en FXML, el scripting será posible y permitirá mejoras considerables en el trabajo realizado. Por ejemplo, se podrían activar y desactivar *capas* de información a función de limitar la visualización a los datos exclusivamente deseados. También se podrían calcular distancias utilizando el Mouse como regla.
- **Fuentes y texto.** Debido a que el correcto transporte de texto en SWF debe de incluir los *glyphs* necesarios para dibujar la fuente, se necesita una investigación más intensiva en la inclusión de dichas curvas a partir de formatos populares como TTF (*True type font*).

3.3.3 Más allá.

Más allá de lo mencionado, es posible trabajar diversas técnicas a fin de alcanzar animación en interacción con botones sobre la película.

También se debe de tener en cuenta que el formato SWF puede tener una compresión utilizando el algoritmo ZLIB, lo cual puede generar archivos más compactos de lo que este trabajo pudo lograr.

Las funciones de Mouse deben de ser pensadas también desde el punto de vista de los dispositivos móviles, donde se cuenta con un *stylus* y no tiene eventos de movimiento, sólo de arrastre y *tapeo.*