Введение в SVG

by Vincent Hardy (Sun Microsystems)

Данная серия документов подготовлена на основе материалов сайта Школы Консорциума W3C. Этот сайт является экспериментальным сервером, на котором содержание документов хранится в формате XML. Пользователям сайта эти документы доступны в виде HTML (преобразование на стороне клиента с помощью таблицы стилей XSLT) и в виде PDF (преобразование тех же документов в XSL-FO, а затем в формат PDF).

Содержание

SVG: общий обзор

Формат **SVG (Scalable Vector Graphics - масштабируемая векторная графика)** - это новый XML-словарь, предназначенный для описания двухмерной векторной графики для интернета и других приложений. В этой статье дается краткий обзор SVG, который будет иллюст-

1: http://xml.nsu.ru/extra/svgintro_1.xml
2: http://xml.nsu.ru/extra/svgintro_2.xml
3: http://xml.nsu.ru/extra/svgintro_3.xml

рироваться примерами графики в формате SVG, а также исходными кодами.

После чтения этой статьи обязательно посетите программу Sun Developer Connection[sm][4], скачайте генератор двухмерной графики SVG Graphics2D SVG Generator[5] и генератор SVG-слайдов SVG Slide Toolkit[6]. Кроме того, познакомьтесь с другими полезными ресурсами по SVG[7].

Происхождение SVG

SVG был разработан Консорциумом W3C[8]. Консорциум W3C - некоммерческая организация, занимающаяся разработкой открытых стандартов, например, HTML и XML, а также многих других важных словарей. В создании SVG участвовало более двадцати организаций, включая Sun Microsystems, Adobe, Apple, IBM и Kodak.

Компания Sun с самого начала принимала участие в разработке спецификации SVG и в рабочей группе экспертов имеется два активных ее представителя.

В настоящее время (21 июля 2000 года) SVG имеет статус рабочего проекта, но ожидается, что вскоре он получит сначала статус кандидата в рекомендации, затем статус предлагаемой рекомендации и, наконец, статус окончательной рекомендации. Вы можете следить за статусом спецификации по SVG на официальном сайте SVG[9].

Преимущества SVG

SVG имеет много преимуществ перед другими форматами изображений, например, перед форматами JPEG и GIF, которые являются наиболее распространенными форматами в интернете. В частности:

- Это **чисто текстовый формат** - файлы SVG могут быть прочитаны и модифицированы широким кругом приложений и обычно меньше по размеру, чем сравнимые изображения в форматах JPEG или GIF.
- **Масштабируемость** - в отличие от растровых форматов GIF или JPEG, SVG является векторным форматом. Это означает, что изображения SVG могут выводиться с высоким качеством при любом разрешении без возникновения эффекта "ступенек", который можно наблюдать при распечатке растровых изображений.
- **Возможность увеличения** - вы можете увеличить любую часть изображения SVG без потери качества.
- **Текст в графике SVG находят поисковики и его можно выделять** - в отличие от растровых изображений, текст в SVG можно выделять и можно находить с помощью поисковых машин. Можно искать заданную строку текста - например, имя города на SVG-карте.
- **Применение скриптов и анимация** - SVG позволяет создавать динамичную и интерактивную графику, гораздо более изощренную, чем можно сделать с помощью растровых изображений или даже Flash-приложений.
- **Совместимость с технологией Java** - SVG совместим с высокотехнологичным Java-графическим движком Java 2D API. Например, познакомьтесь с генератором двухмерной графики SVG[10] в разделе Sun Developer Connection[11].

4: http://wwws.sun.com/software/xml/developers/
5: http://xml.apache.org/batik/
6: http://wwws.sun.com/xml/developers/svg-slidetoolkit
7: http://wwws.sun.com/software/xml/developers/svg/#resources
8: http://www.w3.org/
9: http://www.w3.org/Graphics/SVG
10: http://xml.apache.org/batik/

- Это **открытый стандарт** - SVG является открытой рекомендацией. В отличие от других форматов, SVG не является чьей-либо собственностью.
- Это **чистый XML** - будучи форматом XML, SVG предоставляет все преимущества XML:
  - Возможность работы в различных средах.
  - Интернационализация (поддержка Unicode).
  - Широкая доступность для различных приложений.
  - Легкая модификация через стандартные API, например, DOM API.
  - Легкое преобразование таблицами стилей XSLT.

SVG в интернете

Поскольку SVG является форматом XML, SVG-графика может легко генерироваться веб-серверами "на лету" с использованием стандартных средств XML, многие из которых написаны на Java. Например, веб-сервер может генерировать высококачественные и при этом небольшие по объему графики на основе оперативных биржевых сводок.

SVG позволяет создавать графику с помощью специальных графических пакетов (см. список приложений SVG[12]) или автоматически (например, используя JavaServer Pages[13]). SVG позволяет легко манипулировать графикой с помощью стандартных инструментов XML.

SVG в других средах

Возможность работы в различных средах - ключевое преимущество SVG. Например, с помощью генератора двухмерной графики SVG[14] любое Java-приложение может экспортировать графику в формате SVG. И затем эти изображения могут импортироваться, просматриваться и модифицироваться в различных средах.

Примеры SVG

Замечание

Чтобы посмотреть эти примеры, вам понадобится один из бесплатных просмотрщиков SVG:

- Batik [15] - приложение Java
- Adobe SVG Viewer [16] - для Windows и MacOS

Кстати

Если вы используете для просмотра Adobe SVG Viewer: Нажатие Alt позволяет таскать SVG-графику, Ctrl позволяет увеличивать, а Shift+Ctrl - уменьшать. При нажатии правой кнопки мыши на SVG-графике выпадает меню, в котором одна из опций позволяет просмотреть исходных код SVG-документа.

11: http://wwws.sun.com/software/xml/developers/
12: http://www.w3.org/graphics/svg/svg-implementations
13: http://java.sun.com/products/jsp/
14: http://xml.apache.org/batik/
15: http://xml.apache.org/batik/
16: http://www.adobe.com/svg/viewer/install/

Анимация

Этот пример демонстрирует применение декларативной анимации - вращающийся SVG-текст:

Этот пример демонстрирует анимацию в SVG.
Здесь два анимированных прямоугольника
и анимированная строка текста. Для каждого из
этих объектов элемент <animateTransform>
определяет, как должен трансформироваться объект
(например, вращаться или изменять свой размер).

Оригинальный SVG-файл[17]

Применение скриптов

Этот пример демонстрирует применение скриптов и ссылок. Конкретно, показано, как модифицировать графику в зависимости от событий (кликов мышью) и как можно привязывать к SVG-графике другие документы:

Этот пример демонстрирует применение скриптов и ссылок.
Если вы наведете курсор на букву 'S', 'V' или 'G',
буква станет красной и внизу появится
текстовое описание. Кроме того, это пример создания
ссылки: если вы кликните на букву 'G',
откроется окно и загрузится сайт www.sun.com.
Также здесь имеется пример привязки событий к анимации:
Если вы кликните на букву 'S', анимация запустится.

Оригинальный SVG-файл[18]

17: http://xml.nsu.ru/extra/svgsamples/animation.svg
18: http://xml.nsu.ru/extra/svgsamples/scripting.svg

Фильтры

Этот пример демонстрирует возможности фильтров SVG, как можно создавать текстуры и эффект сияния:

Этот пример иллюстрирует фильтры SVG.
Здесь используется три фильтра:
один создает фоновую текстуру,
один создает эффект сияния букв текста
и еще один создает трехмерный эффект текста.

Оригинальный SVG-файл[19]

Статичная графика

Этот пример демонстрирует смешанное использование различных статичных возможностей SVG: текст, сложные формы, обводки, фильтры (тень), прозрачность и обрезку (clipping):

Этот пример демонстрирует некоторые из возможностей SVG.
Графика содержит несколько сложных форм,
например, расположенные с левой стороны. Кроме того,
здесь используется полупрозрачность (текст "SVG"
наполовину прозрачный), фильтры (тень у текста
"SVG") и тонкое управление текстом
(расстояние между буквами в строках "Scalable",
"Vector" и "Graphics"

Оригинальный SVG-файл[20]

19: http://xml.nsu.ru/extra/svgsamples/filters.svg
20: http://xml.nsu.ru/extra/svgsamples/static.svg

Простые геометрические формы в SVG

В примере этой главы демонстрируются геометрические формы, которые создаются с помощью элементов <rect>, <circle>, <ellipse>, <polygon>, <polyline>, <path>, <g>

Вступление

В SVG имеется несколько основных геометрических форм:

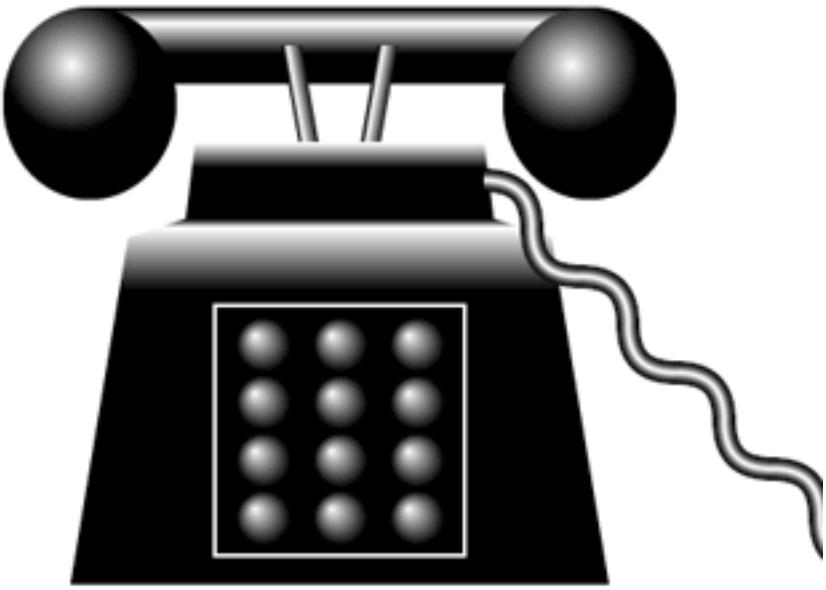
- Прямоугольники
- Окружности
- Эллипсы
- Линии
- Ломаные линии
- Многоугольники

Эти базовые формы, вместе с траекториями, образуют множество графических форм SVG. В этом документе мы познакомимся с тем, как задаются эти формы и как они комбинируются в сложный рисунок (в нашем примере это изображение телефона). Кроме того, здесь будет вкратце описаны методы группировки различных элементов и задание трансформаций внутри группы.

Обратите внимание: стилевые атрибуты не обсуждаются в этой статье. Но вы можете посмотреть в исходном SVG-файле, как они используются для создания градиентных заливок.

Пример

Данное изображение составлено из различных геометрических форм, описанных в формате SVG:



Оригинальный SVG-файл[21]

Простые графические формы образуют изображение телефона. Он состоит из следующих частей:

- Трубка, она состоит из двух эллиптических частей и прямоугольной ручки
- Рычаг, на котором лежит трубка - он образован двумя скошенными прямоугольниками
- Корпус - он изображен с помощью ломаной линии
- Наборного поля - оно изображено с помощью многоугольника и двенадцати круглых кнопок
- Шнур - это траектория, заданная с помощью квадратных кривых Безье

В следующих параграфах мы познакомимся с каждой из этих деталей внимательнее

Прямоугольники и эллипсы

В нашем примере трубка изображена с помощью двух эллипсов и соединяющего их прямоугольника. Следующих код формирует этот прямоугольник и эллипсы:

```
<g id="Handset" transform="translate(70, 90)">
  <rect y="-30" height="40" width="260" />
  <ellipse cy="20" rx="45" ry="50" />
  <ellipse cy="20" rx="45" ry="50" transform="translate(260)" />
</g>
```

Здесь с помощью элемента <g> задается группа объектов с именем "Handset". Начало внутренней системы координат этой группы находится в точке с координатами (100, 60); другими словами, все координаты геометрических форм внутри этого элемента <g> отсчитываются относительно точки с координатами (100, 60) таким образом, что для объектов этой группы эта точка выступает началом координат (0, 0). Мы начинаем с изображения прямоугольника высотой в 20 и шириной 200. Для эллипсов нужно задать дополнительные атрибуты:

- **cx** и **cy** задают центр эллипса

21: http://xml.nsu.ru/extra/svgsamples/shapes.svg

- **rx** задает X-полуось эллипса
- **ry** задает Y-полуось эллипса

Поскольку мы применили элемент <g> для группировки трех составляющих кусочков трубки, мы можем изменить положение трубки на рисунке просто модифицируя значение атрибута transform элемента <g>, для этого нам не нужно изменять координаты каждой элементарной формы.

Следующий пример кода рисует прямоугольники, изображающие рычаг телефона:

```
<g transform="translate(200, 80)">
  <rect x="-30" height="50" width="10" transform="skewX(10)"/>
  <rect x="-30" height="50" width="10" transform="skewX(-10)"
   translate(50, 0)"/>
</g>
```

Эти два прямоугольника немного отличаются от первого, поскольку здесь в атрибуте transform описывается свойство "skew" (скос), которое позволяет нам добиться эффекта скоса стоек рычага. Кроме того, здесь мы сдвигаем прямоугольники, задавая их новые начальные точки. Это делается установкой значений атрибутов x и y элементов <rect>

Окружности, многоугольники и ломаные линии

В нашем примере для изображения корпуса телефона мы используем ломаную линию, для изображения наборного поля - многоугольник, а для изображения кнопок - двенадцать окружностей. Мы группируем наборное поле с кнопками, а затем комбинируем эту группу с корпусом телефона и получаем еще одну группу. Следуя такому групповому подходу, если нам понадобится изменить положение наборного поля с кнопками относительно корпуса телефона, нам нужно будет изменить только координаты внутренней группы, не меняя наборного поля и кнопок. Следующий код рисует окружность с заданным центром:

```
<circle id="phoneKey" cx="0" cy="0" r="14"/>
```

В элементе <circle> атрибуты cx и cy определяют положение центра окружности, а атрибут r задает ее радиус.

Мы познакомились уже с тремя базовыми геометрическими формами - они обладают легко определяемым контуром. Теперь мы переходим к другим формам, которые требуют для описания своего контура более специфичных данных. Например, следующий код задает наборное поле телефона - определяется элемент <polygon>, имеющий четыре угла, он формирует четырехугольник:

```
<polygon points="-65,-20 65,-20 65,110 -65,110" />
```

Важным атрибутом элемента <polygon> является атрибут points. В качестве значения этого атрибута используется список координатных пар точек (два значения для каждой точки). В нашем примере задается прямоугольник, но элемент <polygon> может задавать не только четырехсторонние многоугольники и их стороны могут не быть равными.

Двухмерные фигуры можно задавать и не только как многоугольники. Для этого можно использовать элемент <polyline>. Единственное отличие заключается в том, что элемент <polygon> автоматически закрывает фигуру линией между первой и последней точкой (если они не совпадают), а элемент <polyline> оставляет фигуру открытой. Следующий код рисует корпус телефона как ломаную линию:

```
<polyline points="-75,0 75,0 80,40 110,50 140,230 -140,230 -110,
                  50 -80,40 -75,0" />
```

Траектории

Траектории в SVG представляют контуры объектов. Одним из наиболее важных атрибутов элемента <path> является атрибут d, который содержит данные, описывающие траекторию. Атрибут d несет инструкции типа "moveto" (переместить), "line" (провести линию), "curve" (провести кривую Безье второй или третьей степени), "arc" (провести дугу) и "closepath" (закрыть траекторию). Все эти инструкции обозначаются какой-нибудь одной буквой (например, "moveto" обозначается символом **M**). В нашем примере мы используем инструкции "moveto" и "quadratic bezier curve". Следующий код задает кривую, изображающую телефонный шнур:

```
<path id="cord" d="M 235,130 q 25,0 25,25 q 0,25 25,25 q 25,0 25,
    25 q 0,25 25,25 q 25,0 25,25 q 0,25 25,25 q 25,0 25,25 q 0,
    25 25,25 q 25,0 25,25" />
```

В атрибуте d мы сначала задаем инструкцию **M**, которая дает указание сделать сдвиг к новой точке, от нее начнется кривая. Заглавная **M** в данном случае означает, что в описании используются абсолютные координаты, а маленькая **m** - что используются относительные. После того, как мы с помощью инструкции "moveto" определили начальную точку траектории, мы используем инструкцию **q** для задания сегмента кривой Безье второго порядка. И снова, маленькая **q** означает, что описание дается в относительных координатах, а заглавная **Q** - что в абсолютных. Параметрами команды **q** является серия координатных пар в виде (x1,y1 x,y). Эти координатные пары задают кривую Безье второго порядка, которая проходит из текущей точки в точку с координатами (x,y) используя точку (x1,y1) в качестве контрольной. После выполнения одной инструкции **q**, текущая точка переместится в соответствии с координатным параметром. Поскольку мы используем относительные координаты, мы можем переместить изображение телефонного шнура, просто изменив координаты начальной точки.

Заключение

В примере этой главы мы объединили базовые геометрические формы и траектории. Мы не только использовали самые существенные атрибуты каждого элемента, но и использовали комбинации различных элементов и использовали для них групповые атрибуты. Рисуя траектории, мы использовали относительные координаты, так что перемещать траекторию было несложно. За дополнительной информацией, относящейся к использованию и определению базовых геометрических форм, а также относящейся к различным атрибутам элемента <path>, пожалуйста, обращайтесь на официальный сайт W3C SVG[22].

Работа с текстом в SVG

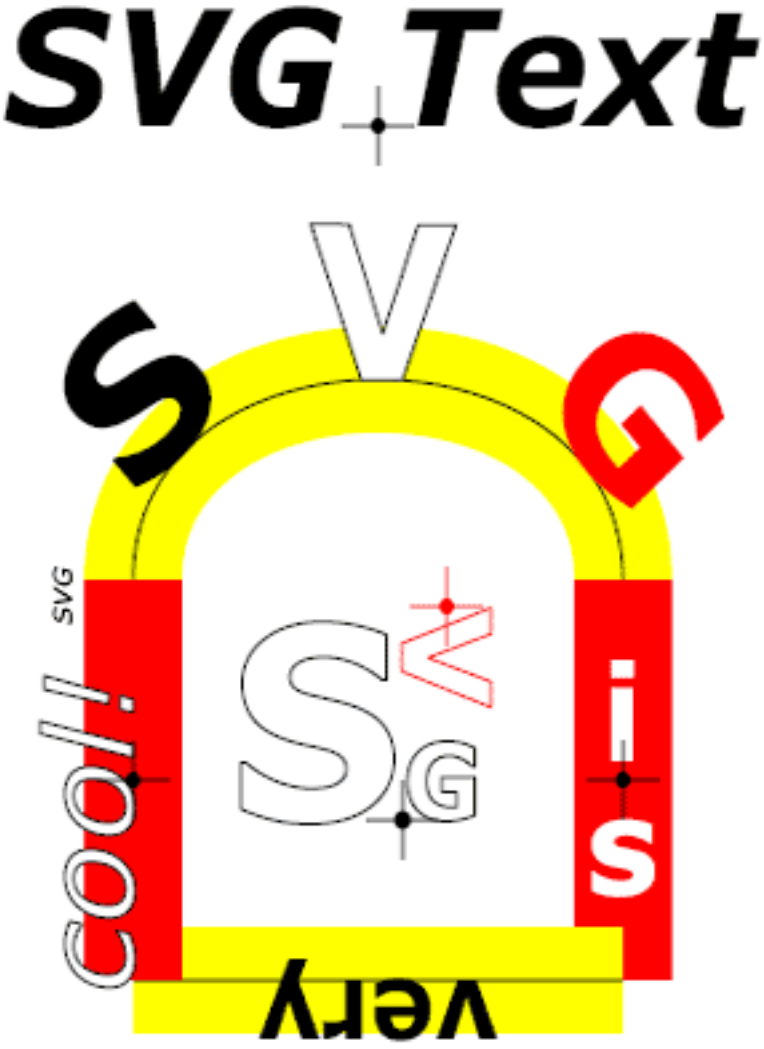
В примере этой главы представляются основные возможности работы с текстом, имеющиеся в SVG, и будут представлены элементы <text>, <tspan>, <textPath>, <defs>, <use>

22: http://www.w3.org/Graphics/SVG

Вступление

В SVG имеются мощные возможности работы с текстом. SVG можно использовать как для задания простой строки текста, так и для определения очень сложного, интенсивно отформатированного текста, или текста, идущего вдоль произвольной траектории. В этой главе мы познакомимся со следующими текстовыми свойствами: определением гарнитуры, ориентацией и направлением текста, выравниванием текста и интенсивным форматированием с помощью элемента <tspan>.

Пример



Оригинальный SVG-файл[23]

В этом примере много текстовых элементов, которые нам сообщают, что "SVG is very cool!":

23: http://xml.nsu.ru/extra/svgsamples/text.svg

- Заголовок ("SVG Text")
- Текст, идущий вдоль кривой ("SVG" черными, белыми и красными буквами)
- Текст, идущий вертикально ("is")
- Перевернутый текст ("very")
- Текст боком ("SVG" как верхний индекс выражения "cool!")
- Внутренний текст SVG ("SVG" с буквой "V" над буквой "G")

В следующих параграфах мы познакомимся с каждым из этих текстовых элементов подробнее

Простой текст SVG

Заголовок "SVG Text" использует элемент <text>, который является родительским элементом для всех текстовых объектов в SVG. Элемент <text> позволяет контролировать положение строки текста, выравнивание текста и другие свойства. Вот код, который отображает наш заголовок:

```
<text x="200" y="80" style="text-anchor:middle; font-size:60;
  font-weight:800; font-family:Verdana; font-style:italic">
  SVG Text</text>
```

Его атрибуты x и y управляют положением "текстового якоря" text-anchor. Якорь является точкой отсчета, относительно которой размещается текст. Атрибут style содержит CSS-свойства, определяющие стиль текста и его положение. Например, свойства font-size, font-family, font-style и font-weight задают характеристики шрифта, используемого для отображения текста. Свойство text-anchor определяет, как позиционируется текст относительно текстового якоря. В нашем примере текст центруется относительно текстового якоря. Далее мы познакомимся и с другими возможными значениями этого свойства.

Мы использовали для описания внешнего вида текста самые простые свойства, но в SVG есть и гораздо более мощные средства управления видом текста.

Обратите внимание: определяя элемент <text>, мы не стали оставлять вокруг текста "SVG Text" пробелов и символов форматирования. Важно знать, что SVG делает с пробелами и символами форматирования внутри элемента <text>. То же самое относится и к элементу <tspan>. За дополнительной информацией обращайтесь к разделу "White space handling" официальной спецификации по SVG.

Текст вдоль траектории

Одна из мощных способностей SVG - это манипулирование текстом, размещаемым вдоль произвольной траектории. В нашем примере для изображения текстовых элементов вдоль кривой линии (это буквы "S","V","G", окрашенные в черный, красный и желтый цвет соответственно) мы использовали следующий код. Тот же метод мы использовали и для отображения других текстовых элементов на соответствующих траекториях:

```
<defs>
  <path id="Path1"
      d="M -100 0 c 0 -100 200 -100 200 0" />
</defs>

<g id="textLayout1" transform="translate(200, 250)">
```

```
  <use xlink:href="#Path1" style="stroke:yellow; stroke-width:40;
   fill:none;" />
  <use xlink:href="#Path1" style="stroke:black; stroke-width:1;
   fill:none;" />

  <text style="font-family:Verdana; font-size:80; font-weight:800;
   fill:blue; text-anchor:middle">
  <textPath xlink:href="#Path1" startOffset="145" >
<tspan style="fill:black">S</tspan>
<tspan style="stroke:black; fill:white;">V</tspan>
<tspan style="fill:red">G</tspan>
      </textPath>
  </text>
</g>
```

Для того, чтобы разместить текстовый блок вдоль границы геометрической формы или траектории, нам сначала нужно определить элемент <path>, используя элемент <defs>. Затем мы должны сконструировать блок текста (он будет размещаться внутри группирующего элемента <g>), который будет отображаться вдоль траектории. Ключевой момент - задание элемента <textPath>, он указывает на нужную траекторию с помощью атрибута xlink:href. Если траектория также должна отображаться, можно использовать запись use xlink:href, как это и сделано в нашем примере. В элементе <textPath> могут быть заданы различные дополнительные атрибуты, позволяющие управлять отображением текста на траектории - кроме тех свойств, которые можно установить для элемента <text>. Атрибут startOffset определяет, в каком месте начинается текст относительно начальной точки траектории.

В этом коде мы использовали элемент <tspan> для окрашивания букв в различные цвета. Смотрите дополнительную информацию о возможностях элемента <tspan> далее в этой главе.

Направление и ориентация текста

Используя такое мощное средство как размещение текста вдоль траектории, мы легко можем размещать текст произвольным образом. Но SVG предоставляет еще и простые средства контроля над направлением текста и его ориентацией.

Следующий код изменяет направление текста "is" (на нашем примере справа) с горизонтального на вертикальное:

```
<text x="100" y="75" style="font-family:Verdana; font-weight:800;
 font-size:50; fill:white; writing-mode:tb;
 glyph-orientation-vertical:0; text-anchor:middle">is</text>
```

Атрибут writing-mode управляет первичным направлением текста. Значение "tb", которое мы здесь использовали, означает направление сверху-вниз. Среди других возможных значений - "rl" (справа-налево) и "tb-rl" (сверху-вниз и справа-налево).

Интересный эффект можно получить, если разместить текст верх ногами. Эффект достигается следующим кодом:

```
<defs>
   <path id="Path3" d="M 100 150 l -200 0" />
</defs>

<g id="textLayout3" transform="translate(200, 250)">
  <use xlink:href="#Path3" style="stroke:yellow; stroke-width:40" />
```

```
  <use xlink:href="#Path3" style="stroke:black; stroke-width:1" />

  <text style="font-family:Verdana; font-size:40; font-weight:900;
   fill:black; stroke:none; text-anchor:middle">
  <textPath xlink:href="#Path3" xml:space="default"
  startOffset="100">very</textPath>
  </text>
</g>
```

Элемент <defs> задает прямую линию, которая идет справа налево, она изменяет направление расположенного на ней текста: текст пишется вверх ногами справа налево. Этот код демонстрирует размещение текста на траектории для изменения его направления.

Элемент <tspan>

В SVG каждый элемент <text> создает одну строку текста. SVG не делает автоматической разбивки строки или переноса слов. Применение элемента <tspan> позволяет разбивать строки внутри текстового блока. Выше мы познакомились с тем, как, используя элементы <tspan>, можно задать различные методы обработки букв, например, придавая им различные цвета.

Следующий код показывает, как можно использовать элемент <tspan> для управления относительным положением внутри текстового блока:

```
<text x="0" y="0" style="font-family:Verdana; font-size:50;
 font-style:italic; fill:white; stroke:black; writing-mode:lr;
 text-anchor:middle;" transform="translate(-100, 75) rotate(-90)">
<tspan dy="0">cool!</tspan>
<tspan dy="-25" style="font-size:10; stroke:none; fill:black;">
 SVG</tspan>
</text>
```

Атрибут dy элемента <tspan> управляет относительным вертикальным положением текущего фрагмента текста ("cool!") относительно начальной точки текстового блока. При установке этого атрибута текущие координаты x-y соответствующим образом изменяются. Устанавливая затем атрибут dy для второго элемента <tspan>, мы получаем эффект верхнего индекса (то есть, "SVG" является верхним индексом для слова "cool!").

Элемент <tspan> важен для позиционирования и управления отображением текста. Он разделяет различные фрагменты текста и применяет к ним различные характеристики положения и отображения.

Выравнивание текста

После того, как мы закончили текст, который идет вокруг нашего изображения, можно приступить ко внутреннему тексту ("SVG" в центре), оформленному с помощью различных свойств выравнивания. Следующий код показывает, как это сделать, используя различные значения атрибута text-anchor:

```
<defs>
  <g id="marker" style="stroke-width:1">
    <line x1="-15" y1="0" x2="15" y2="0"
      style="stroke:currentColor" />
```

```
      <line y1="-15" x1="0" y2="15" x2="0"
        style="stroke:currentColor" />
      <circle cx="0" cy="0" r="3" style="fill:currentColor" />
    </g>
  </defs>

  <g id="textLayout5" transform="translate(180, 290)"
   style="font-weight:800">
    <use xlink:href="#marker" style="fill:black; stroke:black"
      transform="translate(30, 50)" />
    <text x="30" y="50" style="font-family:Verdana; font-size:100;
     stroke:black; fill:none; text-anchor:end">S</text>
    <text x="30" y="50" style="font-family:Verdana; font-size:40;
     font-weight:700; stroke:black; fill:none; text-anchor:start">
     G</text>

    <use xlink:href="#marker" style="color:red"
     transform="translate(48, -30)" />
    <text x="48" y="-30" style="font-family:Verdana; font-size:50;
     stroke:red; fill:none; writing-mode:tb; text-anchor:start">
     V</text>
  </g>
```

После того, как мы сделали первый маркер на заданной позиции с помощью записи use xlink:href, второй и третий элементы <text>, соответствующие буквам "S" и "G", выравниваются горизонтально относительно этого маркера. Элемент <text>, соответствующий букве "S", использует в свойстве text-anchor значение "end", что выравнивает букву "S" по левому краю маркера. Элемент <text>, соответствующий букве "G", использует в свойстве text-anchor значение "start", это выравнивает букву по правому краю маркера. Немного другой метод используется для буквы "V". Свойство элемента <text>, соответствующего этой букве, имеет значение "tb" (то есть, направление сверху вниз). В данном случае атрибут text-anchor имеет другой смысл. Используя значение 'start' свойства text-anchor, данный элемент <text> выравнивается по нижней стороне соответствующего маркера с направлением текста вниз.

Заключение

В этом примере мы познакомились с некоторыми средствами работы с текстом в SVG. Мы можем обрабатывать текст просто как строку текста или сложным образом (например, накладывая текст на сложную траекторию). Мы можем разделить блок текста на несколько фрагментов и применять к каждому различные свойства. Мы можем легко манипулировать направлением текста, его выравниванием и относительным положением. Информацию о других возможностях (например, о разрядке слов и букв) вы можете найти в десятой главе официальной спецификации по SVG, ее можно найти на официальном сайте SVG[24].

24: http://www.w3.org/Graphics/SVG/Overview.htm8