

# The Vector Markup Language

---

The Vector Markup Language (VML), the subject of this chapter, is an XML application that combines vector information with CSS markup to describe vector graphics that can be embedded in Web pages in place of the bitmapped GIF and JPEG images loaded by HTML's `IMG` element. Vector graphics take up less space and thus display much faster over slow network connections than traditional GIF and JPEG bitmap images. VML is supported by the various components of Microsoft Office 2000 (Word, PowerPoint, Excel) as well as by Internet Explorer 5.0. When you save a Word 2000, PowerPoint 2000, or Excel 2000 document as HTML, graphics created in those programs are converted to VML.

## What Is VML?

VML elements represent shapes: rectangles, ovals, circles, triangles, clouds, trapezoids, and so forth. Each shape is described as a path formed from a series of connected lines and curves. VML uses elements and attributes to describe the outline, fill, position, and other properties of each shape. Standard CSS properties can be applied to VML elements to set their positions.

Listing 22-1 is an HTML document. Embedded in this HTML file is the VML code to draw a five-pointed blue star and a red circle. Figure 22-1 shows the document displayed in Internet Explorer 5.0.



### In This Chapter

What is VML?

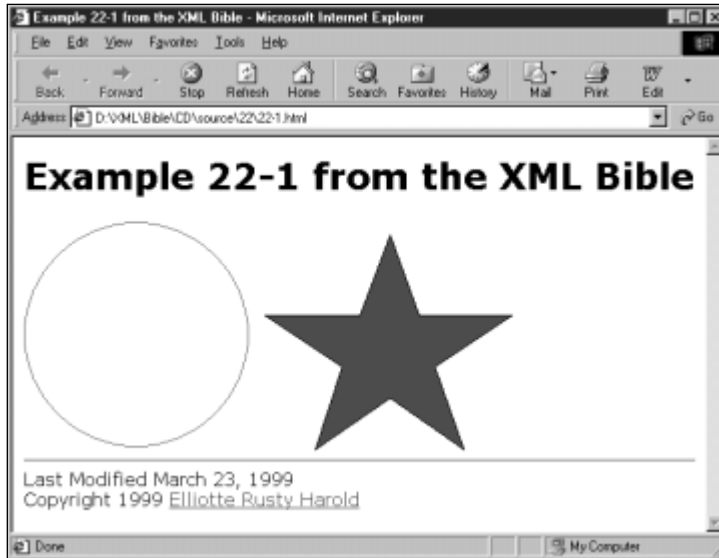
How to draw with a keyboard

How to position VML shapes with CSS properties

VML in Microsoft Office 2000

A quick look at SVG





**Figure 22-1:** An HTML document with embedded VML elements

**Listing 22-1: An HTML document with VML code that draws a five-pointed blue star and a red circle.**

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">
  <head>
    <title>Example 22-1 from the XML Bible</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>
  <body>
    <h1>Example 22-1 from the XML Bible</h1>
    <div>
      <vml:oval
        style="width:200px; height: 200px"
        stroke="true"
        strokecolor="red"
        strokeweight="2">
```

```

</vml:oval>

<vml:polyline
  style="width: 250px; height: 250px"
  stroke="false"
  fill="true"
  fillcolor="blue"
  points="8pt, 65pt, 72pt, 65pt, 92pt, 11pt, 112pt, 65pt,
          174pt, 65pt, 122pt, 100pt, 142pt, 155pt, 92pt,
          121pt, 42pt, 155pt, 60pt, 100pt">
</vml:polyline>
</div>
<hr></hr>
Last Modified March 23, 1999<br />
Copyright 1999
<a href="http://www.macfaq.com/personal.html">
  Elliotte Rusty Harold
</a>
</body>

</html>

```

---

**Listing 22-1** obviously isn't an ordinary HTML document, even though it contains some standard HTML elements. First of all, the `html` root element declares the namespace prefix `vml` as shorthand for `urn:schemas-microsoft-com:vml`. The head element contains an object child with the id `VMLRender`. (`VMLRender` is a program installed with IE5.) And there's a CSS style rule that specifies that all elements in the `vml` namespace (that is, all elements that begin with `vml:`) should have the behavior property `url(#VMLRender)`. This is a relative URL that happens to point to the aforementioned object element. This tells the Web browser to pass all elements that begin with `vml:` (the backslash in `vml\:` is used to ensure that the `:` is treated as part of the element name rather than a selector separator) to the object with the ID `VMLRender` for display.

The body element contains several of the usual HTML elements including `div`, `h1`, `hr`, and `a`. However, it also contains `vml:oval` and `vml:polyline` elements. The `vml:oval` element is set to have a red border (stroke) 2 pixels wide. Furthermore, the `style` attribute sets the CSS `width` and `height` properties of this oval to 200 pixels each. These also set the width and height of the implicit rectangular box that holds the oval. The `vml:polyline` element is set to be filled in blue. The CSS `width` and `height` properties of this oval are set to 250 pixels each. A five-pointed star has ten vertices. The `points` attribute provides 10 pairs of x-y coordinates, one for each vertex.

## Drawing with a Keyboard

Drawing pictures with a keyboard is like hammering a nail into wood with a sponge. Writing VML pictures by typing raw XML code in a text editor is not easy. I suggest you start any attempts to program vector images with some graph paper, and draw the images with a pencil the way you wish to see them on the screen. You can then use the images from the graph paper to determine coordinates for various VML elements like `shape`, `oval`, and `polyline`.

### The `shape` Element

The fundamental VML element is `shape`. This describes an arbitrary closed curve in two dimensions. Most shapes have a path that defines the outline of the shape. The outline may or may not have a stroke with a particular color and width - that is, the outline may or may not be visible. The shape may or may not be filled with a particular color. For example, in Figure 22-1 the circle has a black stroke but no fill, whereas the star has a blue fill but no stroke.

Most properties of a `shape` element can be defined by various attributes. Table 22-1 lists these. For example, here's a `shape` element that draws an isosceles triangle. The `id` attribute gives the shape a unique name. The `coordsize` attribute defines the size of the local coordinate system. The `style` attribute uses CSS `width` and `height` properties to specify the absolute width and height of the box that contains the triangle. The `path` attribute provides data to the `formulas` child element that calculates the exact outline of this particular triangle. And the `fillcolor` attribute makes the entire triangle blue.

```
<vml:shape id="_x0000_t5" coordsize="21600,21600"
  style="width:200px; height: 200px" adj="10800"
  path="m@0,0|0,21600,21600,21600xe" fillcolor="blue">
  <vml:formulas>
    <vml:f eqn="val #0"/>
    <vml:f eqn="prod #0 1 2"/>
    <vml:f eqn="sum @1 10800 0"/>
  </vml:formulas>
</vml:shape>
```

Don't worry if it isn't obvious to you that this is an isosceles triangle. In fact, I'd be surprised if it were even obvious that this is a triangle. Most VML elements (including this one) are drawn using a GUI, and only saved into VML form. Consequently, you don't need to know the detailed syntax of each and every VML element and attribute. However, if you know a little, you can sometimes do some surprising tricks with the VML file that may prove impossible with a graphical editor. For example, you can search for all the blue elements, and change them to red.

**Table 22-1**  
**Attributes of the Shape Element**

<i>Attribute</i>	<i>Default Value</i>	<i>Description</i>
id	none	a unique XML name for the element (same as any other XML ID type attribute)
adj	none	input parameters for formulas child elements that define the shape's path
alt	none	alternate text shown if the shape can't be drawn for any reason; like the ALT attribute of HTML's IMG element
chromakey	none	a transparent background color for the shape which anything behind the shape will show through; for example red or #66FF33
class	none	the CSS class of the shape
coordorigin	0 0	local coordinate corner of the upper left-hand corner of the shape's box
coordsize	1000 1000	width and height of the shape's box in the local coordinate space
fill	true	whether the shape is filled
fillcolor	white	color the shape is filled with; for example red or #66FF33
href	none	URL to jump to when the shape is clicked
opacity	1.0	transparency of shape as a floating point number between 0.0 (invisible) and 1.0 (fully opaque)
path	none	commands that define the shape's path
print	true	whether the shape should be printed when the page is printed
stroke	true	whether the path (outline) of the shape should be drawn
strokecolor	black	color used to draw the shape's path
strokeweight	0.75pt	width of the line used to draw the shape's path
style	none	CSS properties applied to this shape
target	none	the name of the frame loaded when a frameset page loads
title	none	name of the shape
type	none	a reference to the id of a shapetype element
v	none	command defining the path of shape
wrapcoords	none	specifies how tightly text wraps around a shape

Some properties of shapes are more convenient to set with child elements rather than attributes. Furthermore, using child elements allows for more detailed control of some aspects of a shape. For instance, the above isosceles triangle required three formulas to describe its path. Each of these was encapsulated in an `vm1:f` child element. By using attributes, only a single formula could have easily been included. Table 22-2 lists the possible child elements of a shape. If a child element conflicts with an attribute, then the value specified by the child element is used.

Table 22-2  
Shape Child Elements

<i>Element</i>	<i>Description</i>
<code>path</code>	the edge of the shape
<code>formulas</code>	formulas that specify the outline of the shape
<code>handles</code>	visual controls used to alter the shape
<code>fill</code>	how the path should be filled
<code>stroke</code>	how to draw the path, if the artist wants something more elaborate than a straight line and solid color
<code>shadow</code>	the shadow effect for the shape
<code>textbox</code>	the text that should appear inside of the shape
<code>textpath</code>	the vector path used by the text
<code>imagedata</code>	a picture rendered on top of the shape
<code>line</code>	a straight line path
<code>polyline</code>	a path defined by connecting the dots between specified points
<code>curve</code>	a path defined by a cubic Bezier curve
<code>rect</code>	a path defined by a rectangle of a specific height and width
<code>roundrect</code>	a path defined by a rectangle with rounded corners of a specific size
<code>oval</code>	a path defined by an oval enclosed in a rectangle of a specific height and width
<code>arc</code>	a path defined by the arc of an angle between two points
<code>image</code>	a bitmapped image loaded from an external source

Each of these child elements itself has a variety of attributes and child elements used to define its appearance. For instance, `line`, one of the simplest, has `from` and `to` attributes that define the starting and ending points of the line. The value of each of these attributes is a 2-D coordinate in the local coordinate space like `0 5` or

32 10. You can read about the details in the Vector Markup Language submission to the W3C or on the Microsoft Web site at <http://www.microsoft.com/standards/vml/>.



Caution

Treat the VML specification with a pinch of salt. It contains a number of really obvious and quite a few not-so-obvious mistakes.



Note

The line, polyline, curve, rect, roundrect, oval, arc, and image elements do not have to be children of shape. They can stand on their own.

## The shapetype Element

The `shapetype` element defines a shape that can be reused multiple times, by referencing it at a later point within a document using a `shape` element. The `shapetype` element is identical in all ways to the `shape` element except that it cannot be used to reference another `shapetype` element. This element is always hidden. The `shape` element refers to the `shapetype` element using a `type` attribute whose value is a relative URL pointing to the `id` of the `shapetype` element.

For example, Listing 22-2 includes a `shapetype` element that defines a blue right triangle. It also includes three `shape` elements that merely reference this `shapetype`. Thus there are three right triangles in Figure 22-2, even though it's only defined once. Each of these triangles has a different size as set in the individual `shape` elements even though they're all calculated from the same formulas.

### Listing 22-2: Multiple shape elements copy a single shapetype

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">
  <head>
    <title>Example 22-2 from the XML Bible</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>

  <body>
    <h1>Example 22-2 from the XML Bible</h1>

    <vml:shapetype id="fred">
```

*Continued*

## Listing 22-2 (continued)

```

        coordsize="21600,21600"
        fillcolor="blue"
        path="m@0,010,21600,21600,21600xe">
    <vml:formulas>
        <vml:f eqn="val #0"/>
        <vml:f eqn="prod #0 1 2"/>
        <vml:f eqn="sum @1 10800 0"/>
    </vml:formulas>
</vml:shapetype>

<vml:shape type="#fred" style="width:50px; height:50px" />
<vml:shape type="#fred" style="width:100px; height:100px"/>
<vml:shape type="#fred" style="width:150px; height:150px"/>

<hr></hr>
Last Modified March 23, 1999<br />
Copyright 1999
<a href="http://www.macfaq.com/personal.html">
    Elliotte Rusty Harold
</a>
</body>

</html>

```

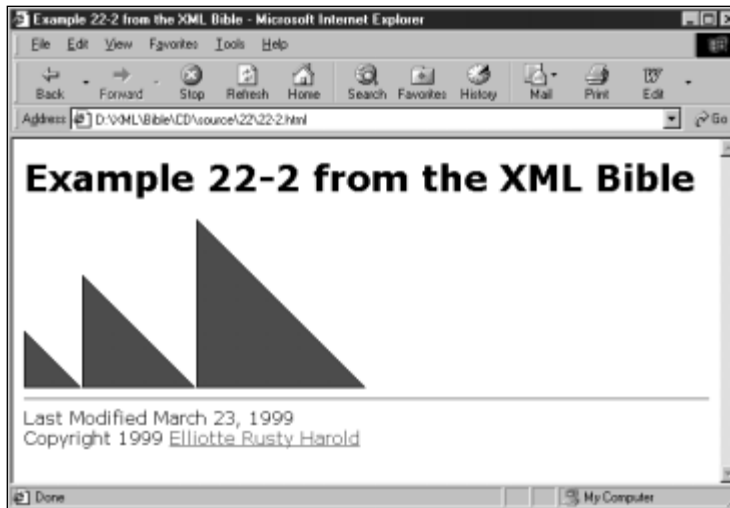


Figure 22-2: Three triangle shapes copied from one shapetype element

When a shape element references a shapetype element, shape may duplicate some of the attributes applied to the shapetype element. In this case, the values associated with shape will override those of shapetype.

## The group Element

The group top-level element combines shapes and other top-level elements. The group has its own local coordinate space in which its child shapes are placed. This entire collection of shapes can then be moved and positioned as a unit. The only attributes the group can have are the core attributes that a shape can have (for example, id, class, style, title, href, target, alt, coordorigin, and coordsize). For example, suppose you need a shape that is a star inside a circle. You can create it by combining an oval with a polyline in a group element like this:

```
<vml:group style="width:6cm; height: 6cm"
  coordorigin="0 0" coordsize="250 250">
  <vml:oval style =
    "position:absolute; top: 15; left: 15; width:200; height: 200"
    stroke="true" strokecolor="black" strokeweight="2"
    fill="true" fillcolor="red">
  </vml:oval>

  <vml:polyline style =
    "position:absolute; top:25; left: 25; width:200; height:200"
    stroke="true" strokecolor="black" strokeweight="5"
    fill="true" fillcolor="blue"
    points="8, 65, 72, 65, 92, 11, 112, 65, 174, 65, 122,
          100, 142, 155, 92, 121, 42, 155, 60, 100">
  </vml:polyline>
</vml:group>
```

The coordsize and coordorigin attributes define the local coordinate system of the elements contained in the group. The coordsize attribute defines how many units there are along the width of the containing block. The coordorigin attribute defines the coordinate of the top left corner of the containing block.

This is an abstract system, not one based in any sort of physical units like inches, pixels, or ems. The conversion between the local units and the global units depends on the height and the width of the group. For example, in the above example the group's actual height and width is 6cm by 6cm, and its coordsize is 250 250. Thus, each local unit is 0.024cm (6cm/250). As the height and width of the group change, the sizes of the contents of the group scale proportionately.

Inside a group, all the CSS properties used to position VML like left and width are given as non-dimensional numbers in the local coordinate space. In other words, unlike normal CSS properties, they do not use units, and are only pure numbers, not real lengths. For example, consider this group:

```
<vml:group style="width: 400px; height: 400px"
```

```
coordsize="100,100"  
coordorigin="-50,-50">
```

The containing block is 400 pixels wide by 400 pixels high. The `coordsize` property specifies that there are 100 units both horizontally and vertically within this group. Each of the local units is four pixels long. The coordinate system inside the containing block ranges from -50.0 to 50.0 along the x-axis and -50.0 to 50.0 along the y-axis with 0.0, 0.0 right in the center of the rectangle. Shapes positioned outside this region will not be truncated, but they are likely to fall on top of or beneath other elements on the page. All children of the group are positioned and sized according to the local coordinate system.

## Positioning VML Shapes with Cascading Style Sheet Properties

VML elements fit directly into the CSS Level 2 visual rendering model, exactly like HTML elements. This means that each VML element is contained in an implicit box, which positions at a certain point on the page. The following standard CSS properties place the box at particular absolute or relative positions on the page:

- ♦ `display`
- ♦ `position`
- ♦ `float`
- ♦ `clear`
- ♦ `height`
- ♦ `width`
- ♦ `top`
- ♦ `bottom`
- ♦ `left`
- ♦ `right`
- ♦ `border`
- ♦ `margin`
- ♦ `visibility`
- ♦ `z-index`



Chapter 12, *Cascading Stylesheets Level 1*, and Chapter 13, *Cascading Style Sheets Level 2*, discuss CSS.

In addition to supporting the standard CSS2 visual-rendering model, VML adds four new properties so shapes can be rotated, flipped, and positioned. These are:

- ♦ rotation
- ♦ flip
- ♦ center-x
- ♦ center-y


**Note**

Personally, I think adding non-standard CSS properties to the `style` attribute is a very bad idea. I would much prefer that these properties simply be additional attributes on the various VML shape elements. The `center-x` and `center-y` properties are particularly annoying because they do nothing the `left` and `right` properties don't already do.

VML elements use a `style` attribute to set these properties, just like HTML elements. This has the same syntax as the HTML `style` attribute. For example, this VML oval uses its `style` attribute to set its position, border, and margin properties:

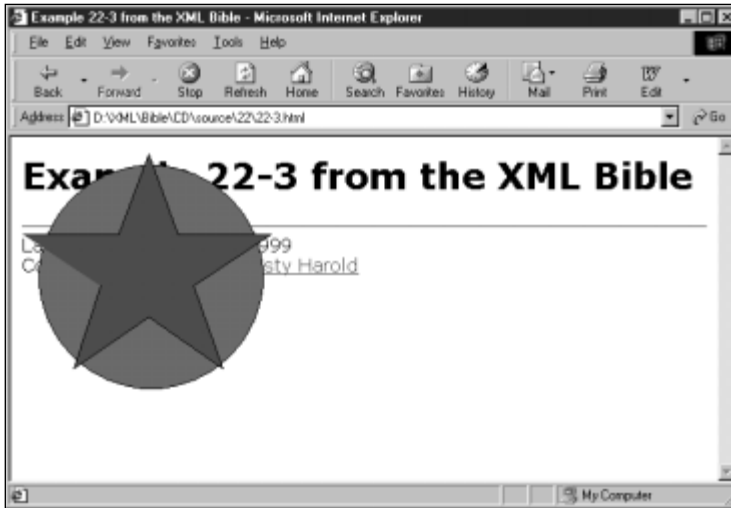
```
<vml:oval style="top: 15; left: 15; width:200; height: 100;
margin:10; border-style:solid; border-right-width: 2;
border-left-width: 2; border-top-width: 1.5;
border-bottom-width: 1.5"
stroke="false" fill="true" fillcolor="green">
</vml:oval>
```

VML shapes are positioned on the page using the CSS `position`, `left`, `right`, `width`, and `height` properties. If the `position` property has the value `absolute`, the invisible rectangular box that contains the shape is placed at particular coordinates relative to the window that displays the shape, regardless of what else appears on the page. This means that different shapes and HTML elements can overlap. VML uses the `z-index` CSS property to layer the first (lowest) to the last (highest) layer, with the latest elements obscuring the earlier elements. This allows you to stack elements on top of each other to build intricate images for your Web pages. If elements don't have `z-index` properties, then elements that come later in the document are placed on top of elements that come earlier in the document.

Listing 22-3 uses absolute positioning to place the blue star on top of the red circle, which is itself on top of the `h1` header and the signature block. Figure 22-3 shows the result.

The default value of the `position` property is `static`, which simply means that both HTML elements and VML shapes are laid out one after the other, each taking as much space as it needs, but none laying on top of each other.

The `position` property can also be set to `relative`, which begins by placing the box where it would normally be, and then offsetting it from that position by the amount specified in the `top`, `bottom`, `left`, and `right` properties.



**Figure 22-3:** A blue star on top of a red circle on top of the body of the page

### Listing 22-3: VML code that draws a five-pointed blue star on top of a red circle

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">

  <head>
    <title>Example 22-3 from the XML Bible</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>

  <body>
    <h1>Example 22-3 from the XML Bible</h1>

    <div>
      <vml:polyline
        style="position:absolute; top:0px; left:0px;
          width: 250px; height: 250px; z-index: 1"
        stroke="false"
        fill="true"
        fillcolor="blue">
```

```

        points="8pt, 65pt, 72pt, 65pt, 92pt, 11pt, 112pt, 65pt,
              174pt, 65pt, 122pt, 100pt, 142pt, 155pt, 92pt,
              121pt, 42pt, 155pt, 60pt, 100pt">
</vml:polyline>

<vml:oval style="position:absolute; top:25px; left:25px;
              width:200px; height: 200px; z-index: 0"
          stroke="false"
          fill="true"
          fillcolor="red">
</vml:oval>

</div>
<hr></hr>
Last Modified March 23, 1999<br />
Copyright 1999
<a href="http://www.macfaq.com/personal.html">
  Elliotte Rusty Harold
</a>
</body>

</html>

```

---

## The rotation Property

The `rotation` property does not exist in standard CSS, but it can be used as a CSS property of VML shapes. The value of the `rotation` property represents the number of degrees a shape is rotated in a clockwise direction, about the center of the shape. If a negative number is given, the object is rotated counterclockwise. Degree values are specified in the format `45deg`, `90deg`, `-30deg`, and so forth. Listing 22-4 rotates Listing 22-1's star by 120 degrees. Figure 22-4 shows the result.

## The flip Property

The `flip` property does not exist in standard CSS, but it can be used as a CSS property of VML shapes. It flips a shape around either its *x*- or *y*-axis, or both. This is given as a CSS property on the `style` attribute of a VML shape element. To flip the *y* coordinates about the *x*-axis, set `flip` to *y*. To flip the *x* coordinates about the *y*-axis, set `flip` to *x*. Listing 22-5 flips the shape about its *x*-axis. Figure 22-5 shows the result.

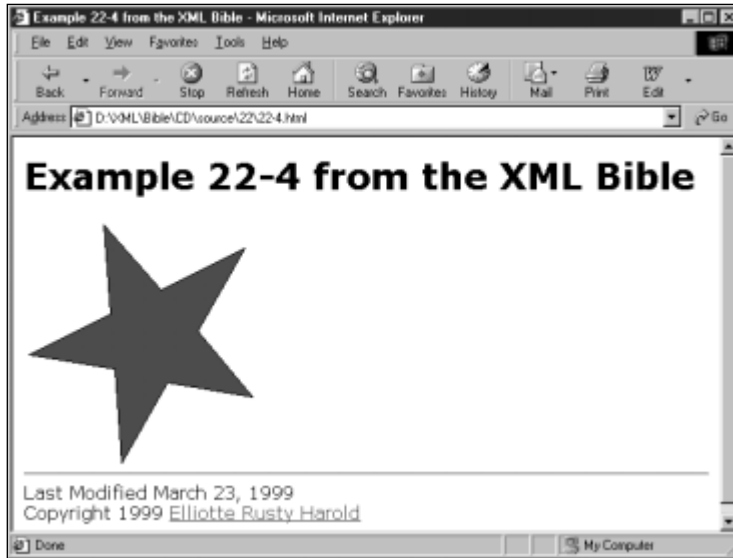


Figure 22-4: A star rotated by 120 degrees

### Listing 22-4: A star rotated by 120 degrees

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">
  <head>
    <title>Example 22-4 from the XML Bible</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>
  <body>
    <h1>Example 22-4 from the XML Bible</h1>
    <div>
      <vml:polyline
        style="width: 250px; height: 250px; rotation: 120deg"
        stroke="true"
        strokecolor="black"
        strokeweight="5"

```

```

fill="true"
fillcolor="blue"
points="8pt, 65pt, 72pt, 65pt, 92pt,11pt, 112pt, 65pt,
       174pt, 65pt, 122pt,100pt, 142pt, 155pt, 92pt,
       121pt, 42pt, 155pt, 60pt, 100pt, 8pt, 65pt">
</vml:polyline>
</div>
<hr></hr>
Last Modified March 23, 1999<br />
Copyright 1999
<a href="http://www.macfaq.com/personal.html">
  Elliotte Rusty Harold
</a>
</body>

</html>

```

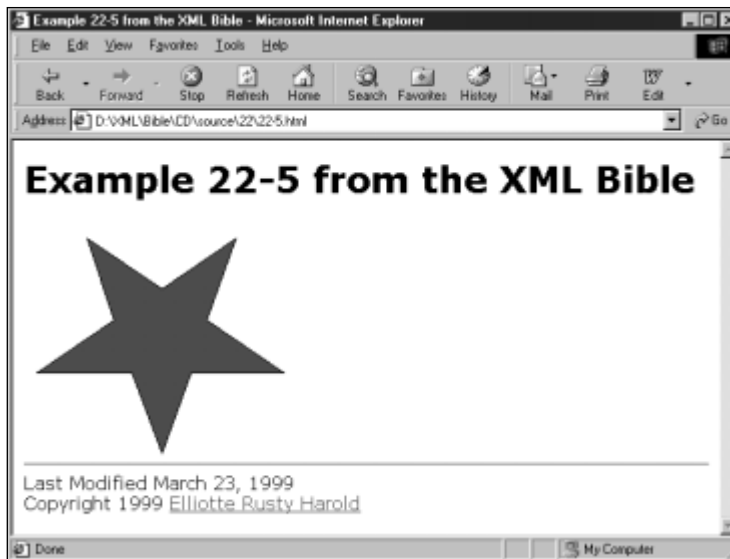


Figure 22-5: The star flipped on its x-axis

### Listing 22-5: A star flipped about its x-axis

```

<html xmlns:vml="urn:schemas-microsoft-com:vml">

  <head>
    <title>Example 22-5 from the XML Bible</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>

  <body>
    <h1>Example 22-5 from the XML Bible</h1>

    <div>

      <vml:polyline
        style="width: 250px; height: 250px; flip: y"
        stroke="true"
        strokecolor="black"
        strokeweight="5"
        fill="true"
        fillcolor="blue"
        points="8pt, 65pt, 72pt, 65pt, 92pt,11pt, 112pt, 65pt,
              174pt, 65pt, 122pt,100pt, 142pt, 155pt, 92pt,
              121pt, 42pt, 155pt, 60pt, 100pt, 8pt, 65pt">
      </vml:polyline>
    </div>
    <hr></hr>
    Last Modified March 23, 1999<br />
    Copyright 1999
    <a href="http://www.macfaq.com/personal.html">
      Elliotte Rusty Harold
    </a>
  </body>

</html>

```

## The center-x and center-y Properties

The `center-x` and `center-y` properties locate the center of the block box that contains the shape. These properties offer alternatives to the `left` and `right` CSS properties and ultimately convey the same information. Because `center-x` and `left` are alternatives for each other as are `center-y` and `right`, you should not specify them both. If you employ both, then the value associated with `center-x` and `center-y` is used.

## VML in Office 2000

Microsoft Word, Excel, and PowerPoint support VML by converting graphics drawn in these programs, into VML markup on HTML pages. In order to do this, you have to set up the Office products correctly.

### Settings

The settings are in essentially the same location in each of the Office components that can create VML. To set VML as the default graphics type, you must perform the following steps:

1. From within Microsoft PowerPoint/Word/Excel, open the Tools menu and select Options.
2. Select the General tab.
3. Click the Web Options button.
4. Select the Pictures tab from the Web Options dialog window.
5. Check the option that reads: “Rely on VML for displaying graphics in browsers,” as shown in Figures 22-6 (PowerPoint), 22-7 (Word), and 22-8 (Excel).

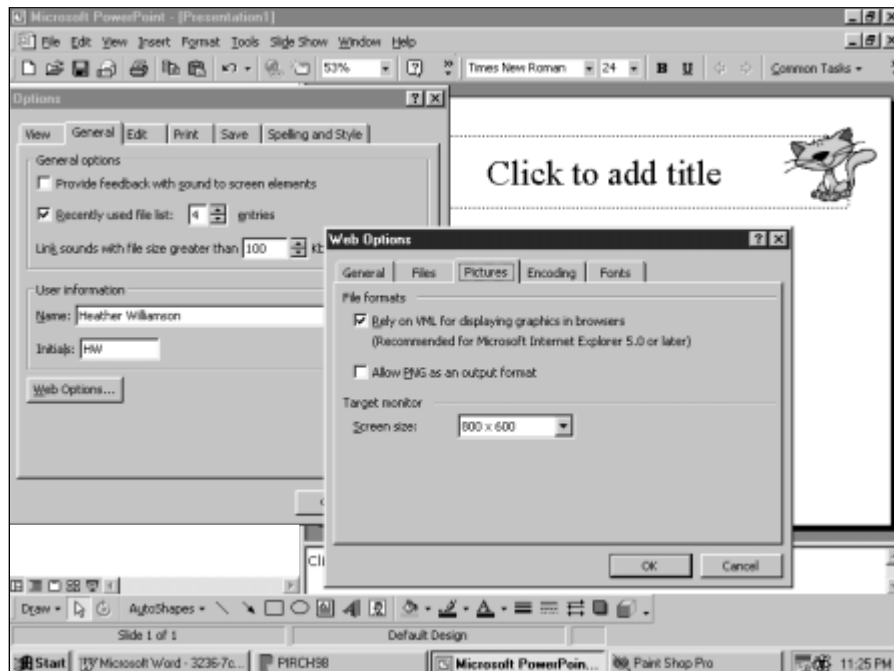


Figure 22-6: Setting VML as the default graphic type in PowerPoint

- Click the OK button on the Web Options window, then OK again on the main program Options window, as shown in Figures 22-6, 22-7, and 22-8. PowerPoint/Word/Excel is now configured to use VML graphics whenever you save a presentation in Web format.

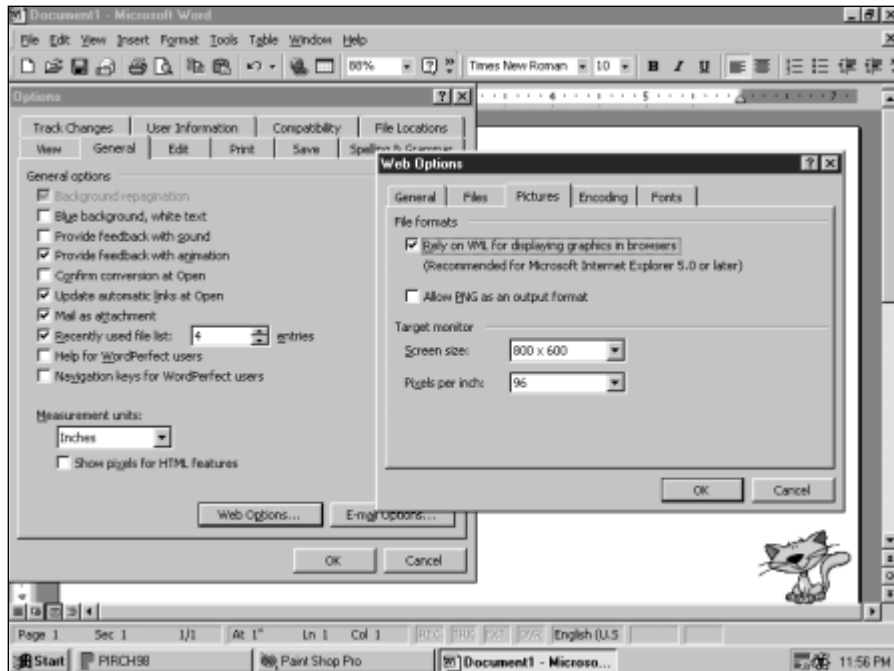


Figure 22-7: Setting VML as the default graphic type in Microsoft Word

Office 2000 will only export into VML those images you drew in their documents using their drawing tools. This means that you cannot use PowerPoint or Word as a conversion utility for other graphics that you have embedded in Office documents.

## A Simple Graphics Demonstration of a House

Office 2000 may not have all the power of Adobe Illustrator or Corel Draw, but it does make drawing simple graphics easy—much easier than drawing with the key-board as shown previously. PowerPoint is the most graphically oriented of the Office components, so let's demonstrate by using PowerPoint to draw a little house. By employing the following steps, it's as simple as drawing a few squares, circles, and triangles:

- Open a new blank presentation from within PowerPoint using the File menu, New option.

2. Select Blank Presentation, and then click OK.
3. In the New Slide window, select the slide with only a title bar at the top, as shown in Figure 22-9, then click OK.

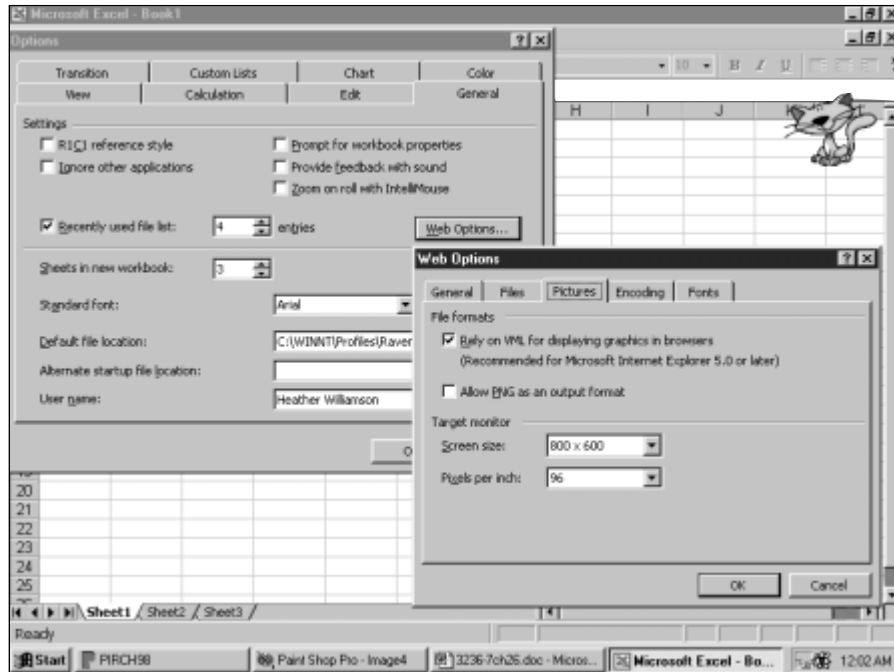


Figure 22-8: Setting VML as the default graphic type in Microsoft Excel

4. Click in the Title bar area, and give your slide a name, for example “My VML House”.
5. On the drawing toolbar at the bottom of the window, click the Rectangle tool. Use this tool to draw the foundation for the VML House.
6. On the drawing toolbar, click the AutoShapes button, select the Basic Shapes option, and then the Isosceles triangle.
7. Draw a roof over the house.
8. Use the Oval and Rectangle tools to draw windows and doors on your house, until your image looks something like the one shown in Figure 22-10.
9. Open the File menu, and select Save As Web Page. Type the name of the page, for example “VMLHouse.html” then click the Save button.

10. Close PowerPoint, and open the file you just created using Internet Explorer 5.0, or select Preview Web Page Preview from the File menu.

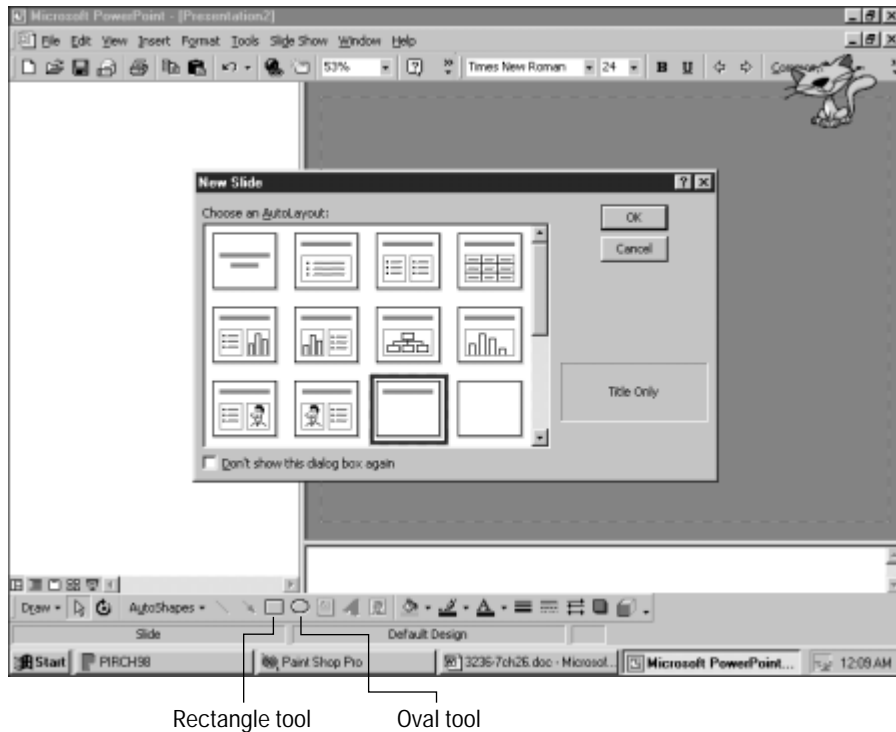


Figure 22-9: Selecting a template for our slide

Figure 22-11 shows the resulting Web page. The HTML and VML code created by PowerPoint to display this slide is shown in Listing 22-6. As well as a lot of standard HTML and VML code, you also see a number of elements in the `urn:schemas-microsoft-com:office:office` and `urn:schemas-microsoft-com:office:powerpoint` namespaces. These contain information that most Web browsers won't use, but that PowerPoint will if the HTML file is opened in PowerPoint. The purpose of these elements is to allow a document to make a round trip from PowerPoint to HTML and back again without losing anything along the way.

Note

The VML house will only be shown in Internet Explorer 5.0 or later. Netscape browsers will only see the embedded images, not the VML.

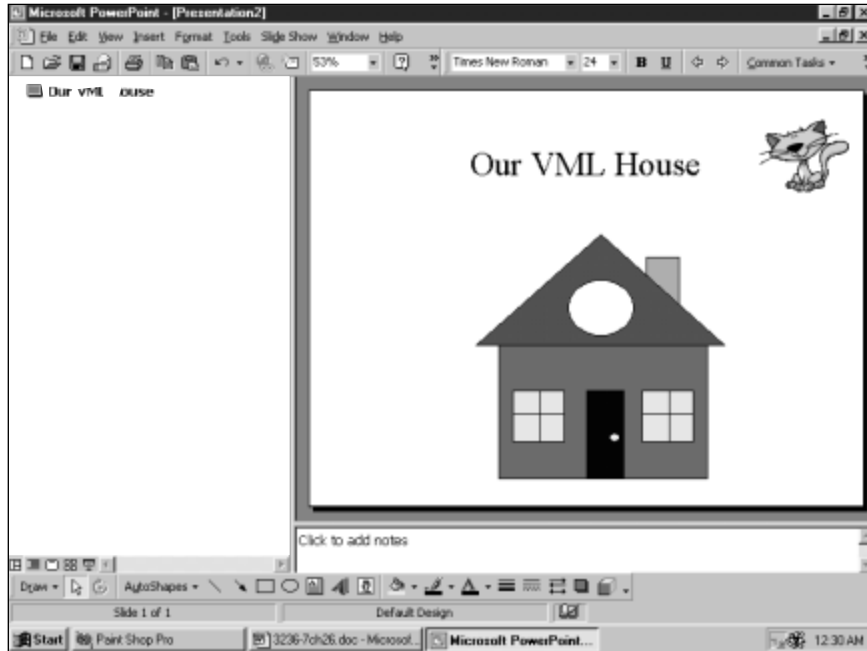


Figure 22-10: The VML House in PowerPoint 2000, ready for conversion into VML text

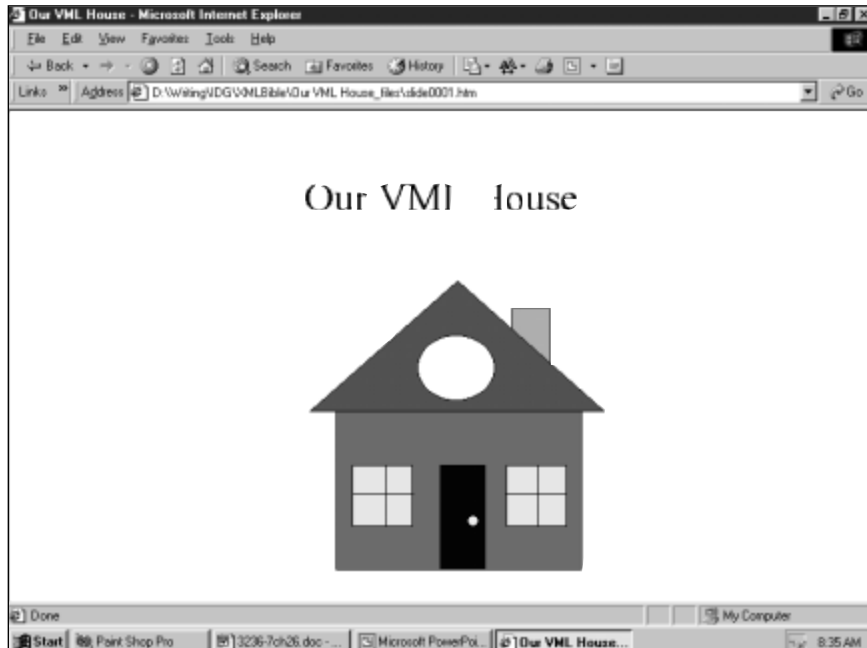


Figure 22-11: The VML House, shown as a Web page, in Internet Explorer 5.0

### Listing 22-6: The “Our VML House” PowerPoint slide converts to an HTML file with embedded VML for use on the Web

```

<html xmlns:v="urn:schemas-microsoft-com:vml"
      xmlns:o="urn:schemas-microsoft-com:office:office"
      xmlns:p="urn:schemas-microsoft-com:office:powerpoint"
      xmlns="-//W3C//DTD HTML 4.0//EN">

<head>
  <meta http-equiv=Content-Type content="text/html;
        charset=windows-1252">
  <meta name=ProgId content=PowerPoint.Slide>
  <meta name=Generator content="Microsoft PowerPoint 9">
  <link id=Main-File rel=Main-File
        href="..\\Our%20VML%20House.htm">
  <link rel=Preview href=preview.wmf>

  <!--[if !mso]>
  <style>
    v\:* {behavior:url(#default#VML);}
    o\:* {behavior:url(#default#VML);}
    p\:* {behavior:url(#default#VML);}
    shape {behavior:url(#default#VML);}
    v\:textbox {display:none;}
  </style>
  <![endif]-->
  <title>Our VML House</title>
  <meta name=Description content="8-Mar-99: Our VML House">
  <link rel=Stylesheet href="master03_stylesheet.css">

  <![if !ppt]>
  <style media=print>
  <!--.sld
    {left:0px !important;
    width:6.0in !important;
    height:4.5in !important;
    font-size:103% !important;}
  -->
  </style>
  <script src=script.js>
  </script>
  <!--[if vml]>
  <script>
    g_vml = 1;
  </script>
  <![endif]-->
  <script for=window event=onload>
  <!--LoadSld( gId );
    MakeSldVis(0);
  //-->
  </script>
  <![endif]>

```

```

<o:shapelayout v:ext="edit">
  <o:idmap v:ext="edit" data="2"/>
</o:shapelayout>
</head>

<body lang=EN-US style='margin:0px;background-color:white'
  onresize="_RSW()">
  <div id=SlideObj class=sld
    style='position:absolute;top:0px;left:0px;
      width:554px;height:415px;font-size:16px;
      background-color:white;clip:
        rect(0%, 101%, 101%, 0%);
      visibility:hidden'>
    <p:slide coordsize="720,540"
      colors="#FFFFFF,#000000,#808080,#000000,#00CC99,#3333CC,
        #CCCCFF,#B2B2B2"
      masterhref="master03.xml">
    <p:shaperange href="master03.xml#_x0000_s1025"/>
    <![if !ppt]>
      <p:shaperange href="master03.xml#_x0000_s1028"/>
    <![if !vml]>
      
    <![endif]>
    <p:shaperange href="master03.xml#_x0000_s1029"/>
    <![if !vml]>
      
    <![endif]>
    <![endif]>
    <v:rect id="_x0000_s2063"
      style='position:absolute;left:438pt;top:3in;
        width:42pt; height:78pt;mso-wrap-style:
          none;v-text-anchor:middle'
      fillcolor="#0c9 [4]"
      strokecolor="black [1]">
    <v:fill color2="white [0]"/>
    <v:shadow color="gray [2]"/>
    </v:rect>
    <p:shaperange href="master03.xml#_x0000_m1026"/>
    <v:shape id="_x0000_s2050"
      type="#_x0000_m1026"
      style='position:absolute;left:54pt;top:48pt;
        width:612pt; height:90pt'>
    <v:fill o:detectmouseclick="f"/>
    <v:stroke o:forcedash="f"/>

```

*Continued*

## Listing 22-6 (continued)

```

    <o:lock v:ext="edit" text="f"/>
    <p:placeholder type="title"/>
  </v:shape>
  <v:rect id="_x0000_s2051"
    style='position:absolute; left:246pt;top:330pt;
      width:270pt;height:174pt;mso-wrap-style:none;
      v-text-anchor:middle'
    fillcolor="red"
    strokecolor="black [1]">
    <v:shadow color="gray [2]"/>
  </v:rect>
  <v:shapetype id="_x0000_t5"
    coordsize="21600,21600"
    o:spt="5"
    adj="10800"
    path="m@0,0l0,21600,21600,21600xe">
    <v:stroke joinstyle="miter"/>
    <v:formulas>
      <v:f eqn="val #0"/>
      <v:f eqn="prod #0 1 2"/>
      <v:f eqn="sum @1 10800 0"/>
    </v:formulas>
    <v:path gradientshapeok="t"
      o:connecttype="custom"
      o:connectlocs="@0,0;@1,10800;0,21600;10800,21600;
        21600,21600;@2,10800"
      textboxrect="0,10800,10800,18000;
        5400,10800,16200,18000;
        10800,10800,21600,18000;
        0,7200,7200,21600;
        7200,7200,14400,21600;
        14400,7200,21600,21600"/>

    <v:handles>
      <v:h position="#0,topLeft" xrange="0,21600"/>
    </v:handles>
  </v:shapetype>
  <v:shape id="_x0000_s2053"
    type="#_x0000_t5"
    style='position:absolute;left:3in;top:186pt;
      width:324pt;height:2in;mso-wrap-style:none;
      v-text-anchor:middle'
    fillcolor="#33c [5]"
    strokecolor="black [1]">
    <v:shadow color="gray [2]"/>
  </v:shape>
  <v:oval id="_x0000_s2054"
    style='position:absolute;left:336pt;top:246pt;
      width:84pt;height:1in;mso-wrap-style:none;
      v-text-anchor:middle'
    fillcolor="white [0]"
    strokecolor="black [1]">

```

```

    <v:shadow color="gray [2]"/>
</v:oval>
<v:rect id="_x0000_s2055"
  style='position:absolute;left:264pt;top:390pt;
        width:66pt;height:66pt;mso-wrap-style:none;
        v-text-anchor:middle'
  fillcolor="#6ff"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:rect>
<v:rect id="_x0000_s2056"
  style='position:absolute;left:5in;top:390pt;
        width:48pt;height:114pt;mso-wrap-style:none;
        v-text-anchor:middle'
  fillcolor="black [1]"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:rect>
<v:rect id="_x0000_s2057"
  style='position:absolute;left:6in;top:390pt;
        width:66pt;height:66pt;mso-wrap-style:none;
        v-text-anchor:middle'
  fillcolor="#6ff"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:rect>
<v:line id="_x0000_s2058"
  style='position:absolute'
  from="300pt,390pt"
  to="300pt,456pt"
  coordsize="21600,21600"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:line>
<v:line id="_x0000_s2059"
  style='position:absolute'
  from="264pt,420pt"
  to="330pt,420pt"
  coordsize="21600,21600"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:line>
<v:line id="_x0000_s2060"
  style='position:absolute'
  from="468pt,390pt"
  to="468pt,456pt"
  coordsize="21600,21600"
  strokecolor="black [1]">
  <v:shadow color="gray [2]"/>
</v:line>
<v:line id="_x0000_s2061"
  style='position:absolute'

```

*Continued*

## Listing 22-6 (continued)

```

        from="6in,420pt"
        to="498pt,420pt"
        coordsize="21600,21600"
        strokecolor="black [1]">
    <v:shadow color="gray [2]"/>
</v:line>
<v:oval id="_x0000_s2062"
    style='position:absolute;left:390pt;top:444pt;
        width:12pt;height:12pt;mso-wrap-style:none;
        v-text-anchor:middle'
    fillcolor="yellow"
    strokecolor="black [1]">
    <v:shadow color="gray [2]"/>
</v:oval>
<![if !vml]>
    
<![endif]>
<div v:shape="_x0000_s2050" class=T
    style='position:absolute;top:13.01%;
        left:8.48%;width:83.21%;height:9.15%'>
    Our VML House
</div>
</p:slide>
</div>
</body>
</html>

```

## A Quick Look at SVG

Let's be honest about something here. VML may well be another Microsoft technology that follows in the footsteps of ActiveX and Bob; that is, a technology, which will be implemented by Microsoft and no one else. VML is not supported by Netscape Navigator nor is it likely to be.

The W3C has received four different proposals for vector graphics in XML from a wide variety of vendors. It's formed the Scalable Vector Graphics (SVG) working group composed of representatives from all these vendors to develop a single specification for an XML representation of Scalable Vector Graphics. When SVG is complete it should provide everything VML provides plus a lot more including animation, interactive elements, filters, clipping, masking, compositing, and pattern fills. However, both a full SVG specification and software that implements the specification is some time away. VML is here today.

The World Wide Web Consortium released the first working draft of SVG in February of 1999, and revised that draft in April 1999. Compared to other working drafts, however, it is woefully incomplete. It's really not much more than an outline of graphics elements that need to be included, without any details about how exactly those elements will be encoded in XML. I wouldn't be surprised if this draft got pushed out the door a little early to head off adoption of competing efforts like VML.

Microsoft has stated publicly that they intend to ignore any Web graphics efforts except VML. However, they are represented on the SVG working group. Whether its representatives actually participate or whether Microsoft's name is only on the masthead as a political gesture is unknown. In either case, SVG is a development that we are all going to have to watch for the next few development cycles to see where it is going, who is following, and who is leading.

## Summary

In this chapter, we looked at developing VML graphics for use with Internet Explorer 5.0. In addition to the general overview of VML, we specifically looked at:

- ♦ What VML can do for Web graphics.
- ♦ The various elements and attributes associated with VML shapes, and how to use them to create the visual images that you need.
- ♦ How to configure Microsoft Office 2000 applications to use VML when creating graphics for Web documents and presentations.
- ♦ How to draw VML figures using PowerPoint 2000.
- ♦ How SVG may affect Web graphics, and what could happen to VML as a result.

The last few chapters, this one included, have looked at a variety of XML applications designed by third parties. In the next chapter, we will design a new XML application from scratch that covers genealogy.



