Paths

- Paths are described using the following data attributes:
- "moveto" (set a new current point),
- "lineto" (draw a straight line),
- "curveto" (draw a curve using a cubic Bezier),
- "arc" (elliptical or circular arc), and
- "closepath" (close the current path by drawing a line to the last "moveto" point).

- The following example specifies a path in the shape of a triangle.

- The "M" indicates a "moveto," "L"s indicate "lineto"s, and the "z" indicates a "closepath."

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/
svg-20000303-stylable.dtd">
<svg xml:space="preserve" width="5.5in" height="2in">
<path d="M 50 10 L 350 10 L 200 120 z"/>
</svg>
```

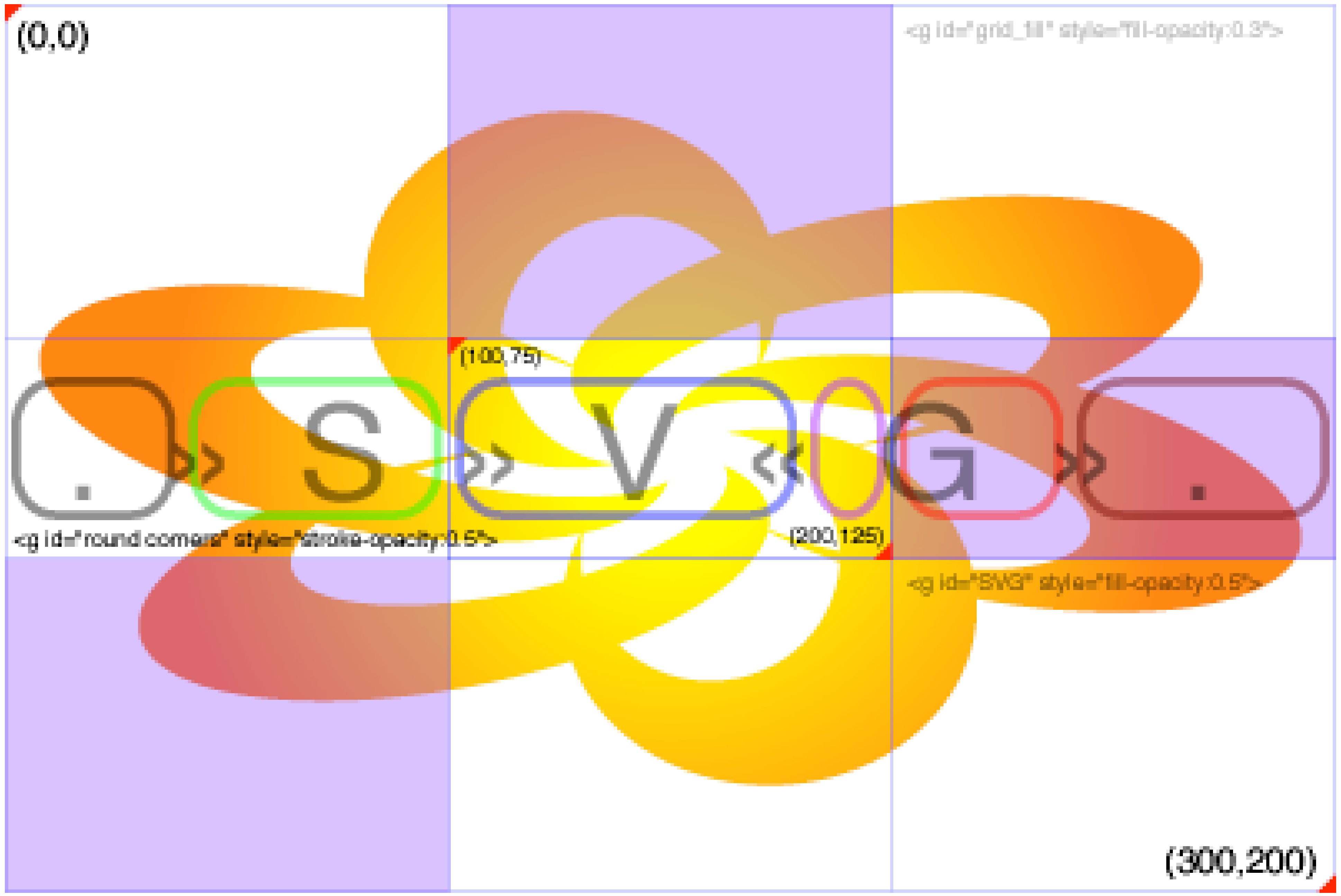
The bounding box

- After the standard language and document definitions, the <svg> element appears.
- The <svg> element defines the SVG document, and can specify, among other things, the user coordinate system, and various CSS unit specifiers.
- The line <svg xml:space="preserve" width="3in" height="2in" viewBox="0 0 300 200"> specifies:
- 1) that whitespace within text elements will be preserved,
- 2) the "intrinsic" width and height of the SVG document — particularly important for specifying print output size, and
- 3) the "bounding box" for the elements of the graphic.

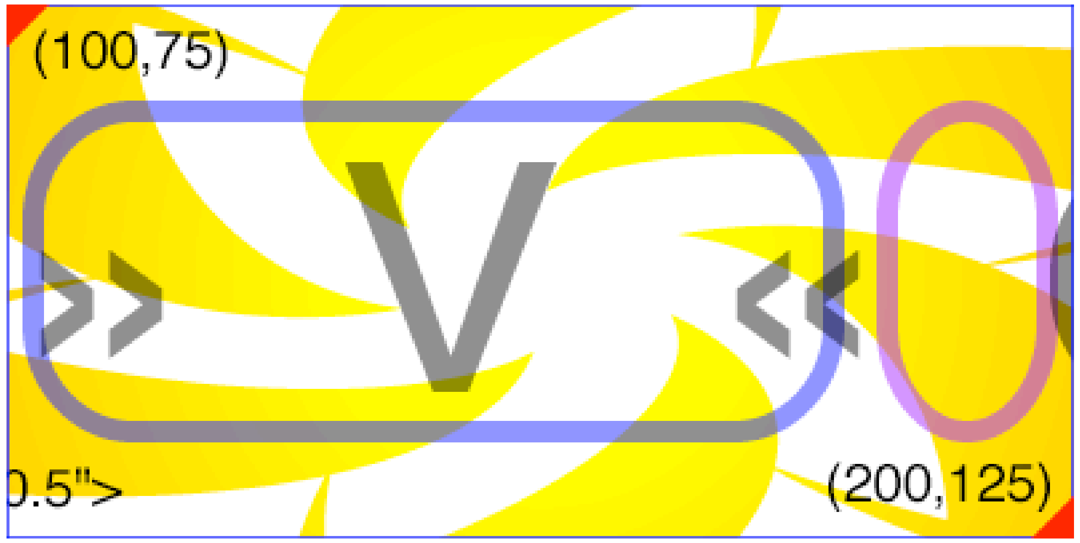
Example

- the SVG file is establishing the boundaries of the image, by stating that in a rectangular space, the upper lefthand x, y coordinates are to be set as 0, 0 and the width and height of the viewing area are 300 and 200.

- `<svg xml:space="preserve" width="3in"  height="2in" viewBox="0 0 300 200">`


(0,0)
<g id="grid_fill" style="fill-opacity:0.3">
(100,75)
S
V
G
<g id="round corners" style="stroke-opacity:0.5">
(200,125)
<g id="SVG" style="fill-opacity:0.5">
(300,200)

The "viewBox" attribute defines the 2D space for the enclosed element. By making the bounding box area smaller, the image can be "cropped" or "pre-zoomed." If the above numbers were changed to 100 75 100 50, the image would look like this:



```
<svg xml:space="preserve" width="100"
height="50" viewBox="100 75 100 50">
```

Graphic objects

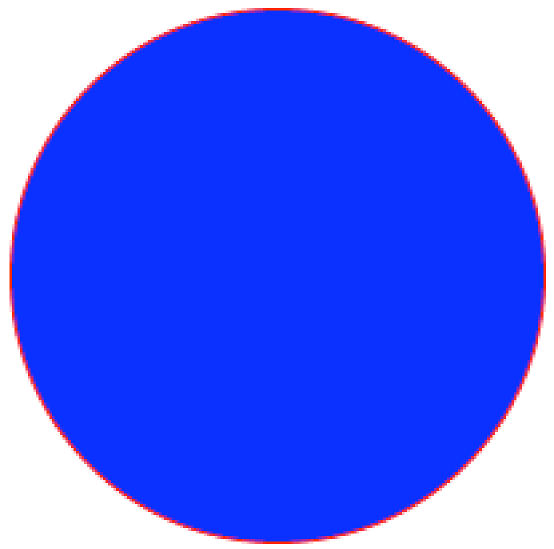
- There are predefined graphic objects such as
  - `<rect/>`
  - `<circle/>`
  - `<ellipse/>`
  - `<polyline/>`
  - `<polygon/>`

- Because a key optimization technique for SVG is in cutting down character count in the file, substituting path data with predefined objects can be very helpful.

For example, the two samples below describe the same blue circle with a red outline

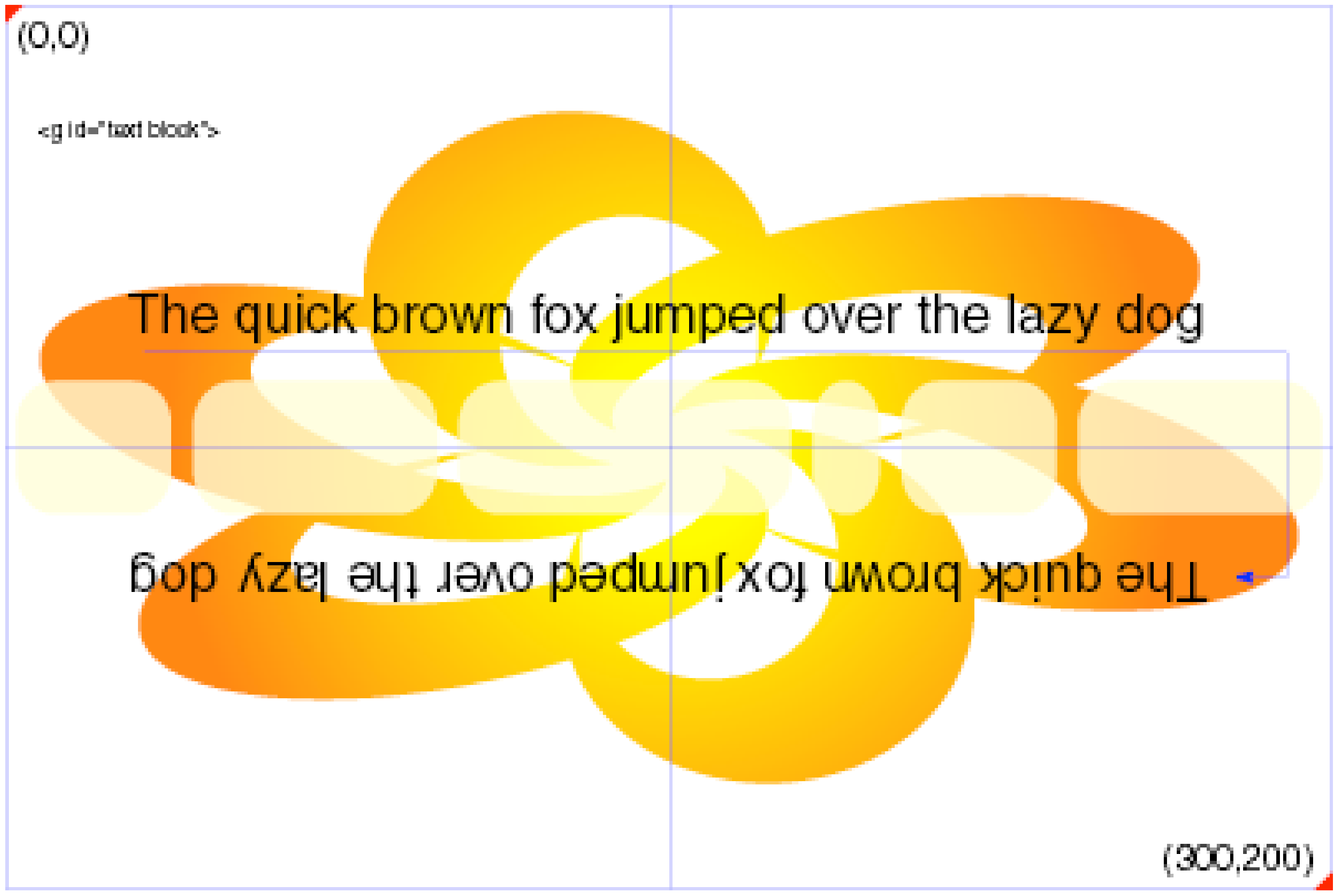
```
<circle style="fill:blue;stroke:red;" cx="200" cy="200" r="100"/>

<path style="fill:blue;stroke:red;" d="M 200,100 c 0,55.23
-44.77,100 -100,100 S 0,155.23,0,100 S 44.77,0,100, 0 s
100,44.77,100,100 z"/>
```



Text

- SVG text elements often resemble straightforward text elements in HTML.
- Font styles and colors are defined as style properties.
- As long as there is no custom kerning or binding to a path within the text string, the text is kept together inside these tags, just as in HTML.
- This allows for very simple edits to text content, as well as easy scriptability for dynamic text generation and manipulation.
- Text properties also include position coordinates that define the relative position of the text element within the bounding box.


(0,0)
<g id="text block">
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
(300,200)

SVG images are not limited to fonts installed in the user's system. Illustrator 9 can convert Type 1 and TrueType fonts to CEF (Compact Embedded Font) fonts that can be embedded within the SVG file.

The example below includes an embedded font.

zoom
in on
THIS
this
is
svg

With the capacity to embed Type 1 fonts, SVG images contain not only searchable and customizable text content, but text rendered and printable in any Type 1 font the designer wants.

Cascading Style Sheets

- Cascading Style Sheets (CSS) are used in Web page design as efficient and effective methods of defining and organizing styles for text and graphics displays on a Web page or entire site.
- SVG supports CSS2 definition style sheets and so can be authored using inline, embedded, and external style sheets.
- Everything from text attributes to layout (as in line spacing) and graphic attributes (fill and stroke properties) can be defined as styles

In the sample image above, without style sheets, every rounded rectangular path would be defined as below:

```
<g id="round_corners">
        <g style="stroke:#990000;stroke-width:2;>
                <path d="M37.693,104.875c0,..."/>
```

In the sample image above, without style sheets, every rounded rectangular path would be defined as below:

```
<g id="round_corners">
    <g style="stroke:#990000;stroke-width:2;>
        <path d="M37.693,104.875c0,..."/>
```

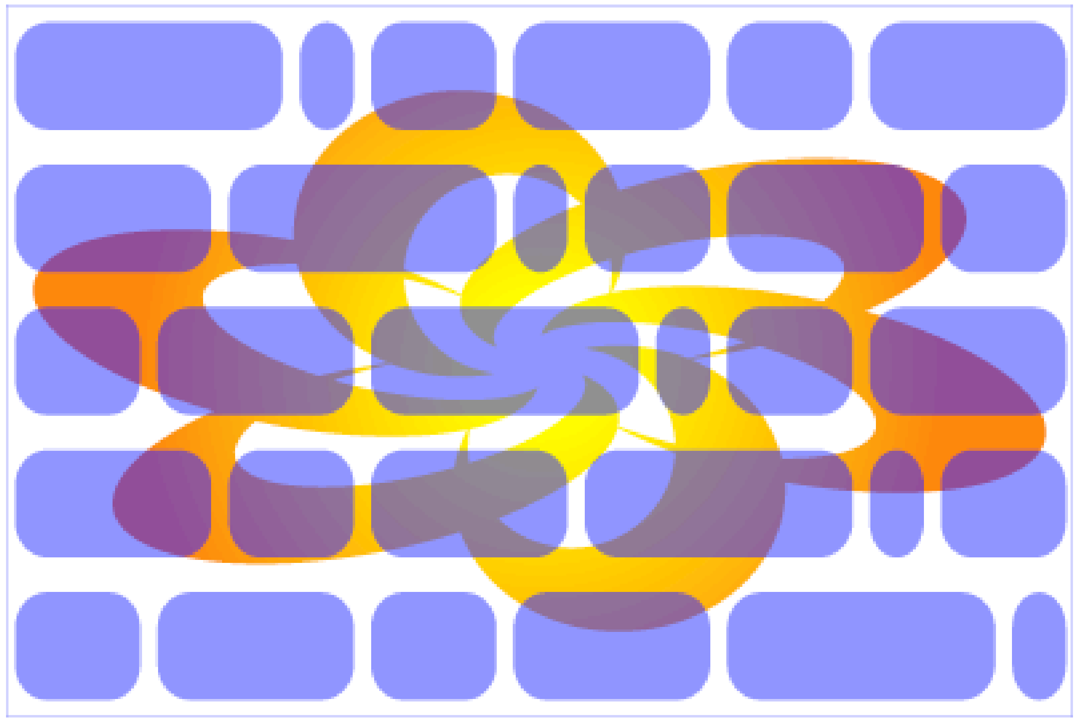
With an embedded style sheet, however, the inline style attribute would be replaced by a "class" attribute that would refer to a style defined once at the top of the document.

Now, if you want to implement a change to the "round_corner" elements, you can simply alter the style definition, and see the change cascade through the document (in this case all 30 elements).

```
.st2{fill:none;stroke:#990000;stroke-width:2;}
```

modified to:

```
.st2{fill:#0000FF;fill-opacity:.5;stroke:none;}
```



Inlined styles

- The advantage to inline styles is improved display performance — images load faster.
- The example above of a "cascading" edit to a style is still applicable to Illustrator-exported SVG.
- The entity definitions are still at the top of the document, and editing these definitions produces the same result as editing a selector's style.
- It should also be noted that using either entities or embedded style sheets also cuts down in word count, and consequently file size.

CSS

- However, SVG files can also refer to styles in external style sheets, which can be shared by the parent HTML document.
- Fonts can be platform-specific, and colors could be tied to a central definition ensuring consistency throughout a Web site
- -- you could just drop an SVG file into a Web site and it would take on the major attributes of the site's design automatically!

Properties and values (fill)

fill

example:

<path style="fill:white">

values: none, current-color, <color>

The "fill" property determines whether an element has a fill or not, and if so, what color. "Current-color" will return the color value specified in a parent document, such as the XML file the SVG resides in.

fillrule

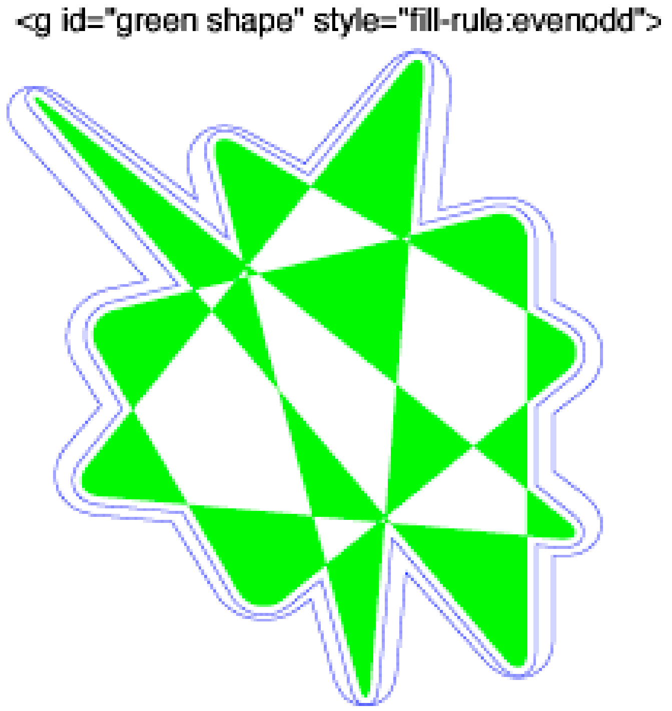
example:

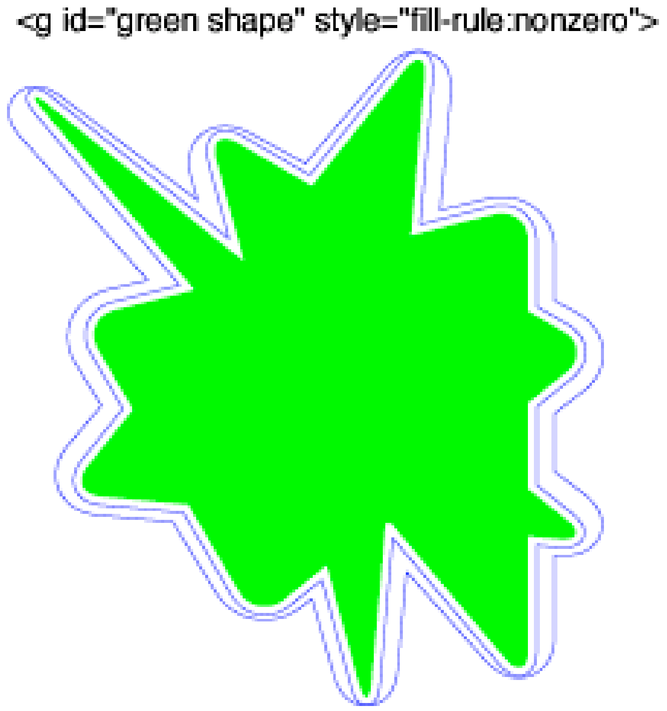
<path style="fillrule:nonzero">

values: evenodd, nonzero, inherit

The "fillrule" property determines whether a path that intersects itself or a compound path (like a donut) will fill only non-adjacent areas, or all areas.

Below are two samples with "evenodd" and "nonzero" fillrules.





"Inherit" means that the value of this element should be the same as the value of the element (or group) enclosing it.

fill-opacity

example:

```
<path style="fill-opacity:0.25">
```

values: any value between 0 and 1

The "fill-opacity" property determines whether an element is solid or transparent (value of 1 is completely opaque, and 0 is completely transparent). There are also the additional opacity properties of "stroke-opacity" and "opacity". While "fill-opacity" refers to the element's fill, "stroke-opacity" refers to its stroke, and "opacity" refers to the opacity of the element as a whole.

Properties and values (strokes)

stroke

example:

```
<path style="stroke:white">
```

values: none, current-color, <color>

The "stroke" property determines whether an element has a stroke or not, and if so, what color. "Current-color" will return the color value specified in a parent document such as the XML file in which the SVG resides.

stroke-width

example:

```
<path style="stroke-width:2">
```

values: <width>, inherit

The "stroke-width" property determines the width of the stroke in user units with 1 as the default. Percentages can also be specified, where the percent of the SVG viewport is applied as the value.

stroke-linecap

example:

```
<path style="stroke-linecap:square">
```

values: butt, round, square, inherit

The "stroke-linecap" property determines the shape to be used at the end of open paths. "Butt" is the default.

stroke-linejoin

example:

```
<path style="stroke-linejoin:bevel">
```

values: miter, round, bevel, inherit

The "stroke-linejoin" property determines the shape to be used at corners of paths. "Miter" is the default.

stroke-miterlimit

example:

```
<path style="stroke-miterlimit:4">
```

values: <miterlimit>, inherit

The "stroke-miterlimit" property determines the how far the miter extends where two line segments meet at a sharp angle. The value must be greater than 1, and corresponds to the values seen in Illustrator's "stroke options: miterlimit". "8" is the default value.

stroke-dasharray

example:

```
<path style="stroke-dasharray:12 12">
```

values: <dasharray>, inherit

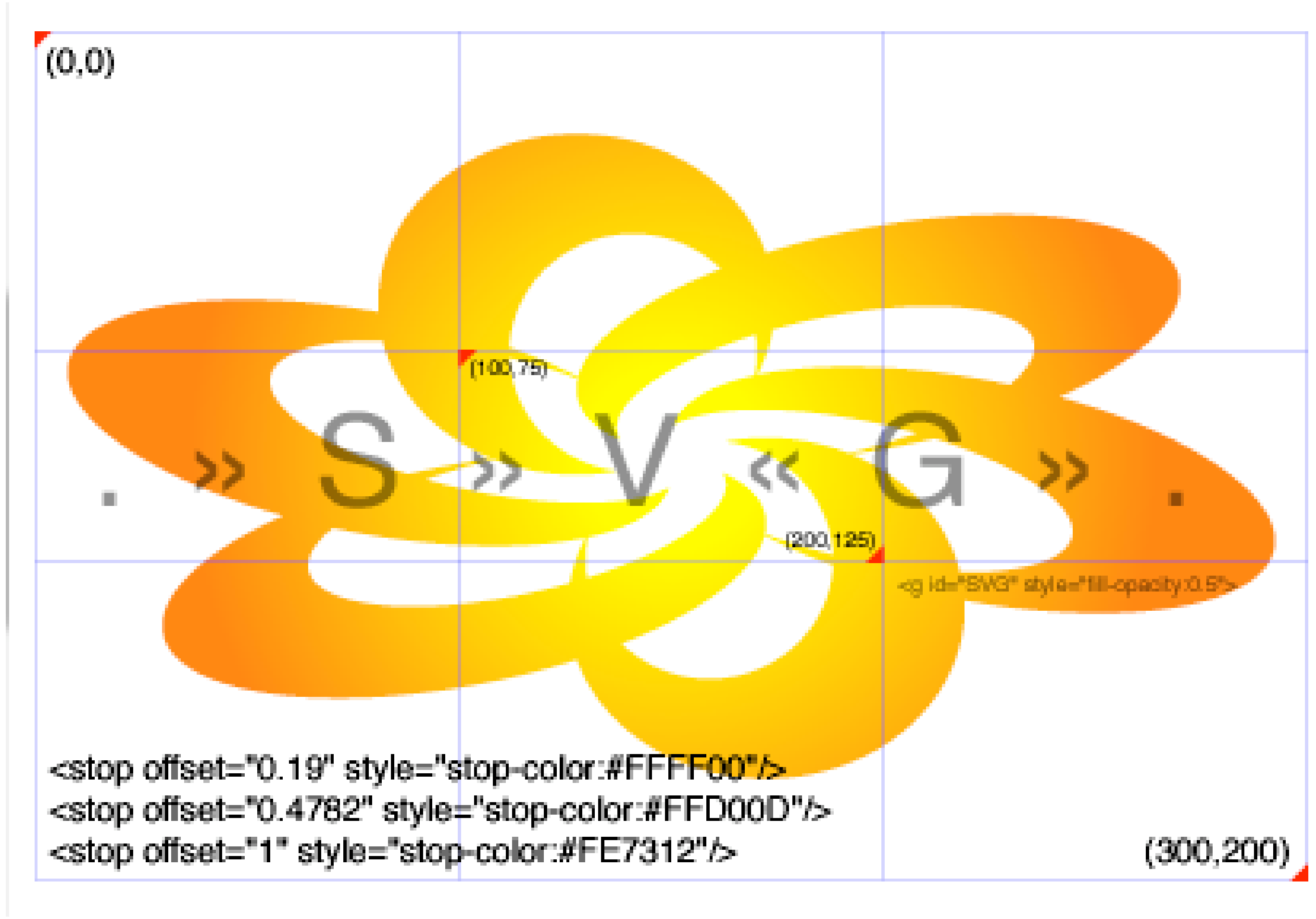
The "stroke-dasharray" property determines the pattern of dashes and gaps for a dashed line.

Gradients

- Gradients are a special case of the "fill" and "stroke" properties.
- Gradients consist of continuously smooth color transitions along a vector from one color to another, possibly followed by additional transitions along the same vector to other colors.
- SVG provides for both linear and radial gradients.
- Gradients are specified within a <defs> element and are then referenced using fill or stroke properties on a given graphics object.
- (A <defs> element is an undrawn element containing definitions, which are referenced later in the SVG document.)
- Whatever is in the <defs> element is not drawn until it is referenced in an item that is to be drawn.

```
<defs>
<radialGradient id="gradient1" cx="150.0005" cy="100"
r="113.7914" fx="150.0005" fy="100"
gradientUnits="userSpaceOnUse">
```

Here, the element named "gradient_star" has a radial gradient (called "gradient1") that goes from yellow to orange



```
<g id="SVG" style="fill-opacity:0.5">
<stop offset="0.19" style="stop-color:#FFFF00"/>
<stop offset="0.4782" style="stop-color:#FFD00D"/>
<stop offset="1" style="stop-color:#FE7312"/>
```

Filter effects

- Vector graphics have the advantage of being quick to download because rendering is done on the client-side.
- However, most designers resort to using bitmap formats in order to produce rich, textured and dimensional-looking graphics.
- Among the many disadvantages of this approach is the general difficulty of keeping the raster data in sync with the rest of the Web site.
- The designer must return to the source image editor to simply change the title of a button, and then must re-export the file for Web use.

Filter Effects

- SVG's filter effects enable designers to create vector graphic images that display using a variety of popular filter effects.
- This capability permits the production of Web artwork in such a way that client-side generation and alteration can be performed easily.
- This means that content for your vector artwork can be generated dynamically, and then rendered with sophisticated filter effects.
- You now get the best of many worlds with compact vector graphics, scriptable text content, and rendered filter effects.

- Filter effects are defined by a <filter> element with an associated ID. Filter effects are applied to elements that have a "filter" property that reference a <filter> element.

  ```
  <?xml version="1.0" standalone="no"?> <!
  DOCTYPE svg PUBLIC "-//W3C//DTD SVG 03March
  2000//EN" "http://www.w3.org/Graphics/SVG/
  SVG-19991203.dtd">

  <svg width="4in" height="3in">
  <defs><filter id="CoolTextEffect">
  <!-- Definition of filter goes here -->
  </filter></defs>
  <text style="filter:url(#CoolTextEffect)">Text
  with a cool effect</text>
  </svg>
  ```

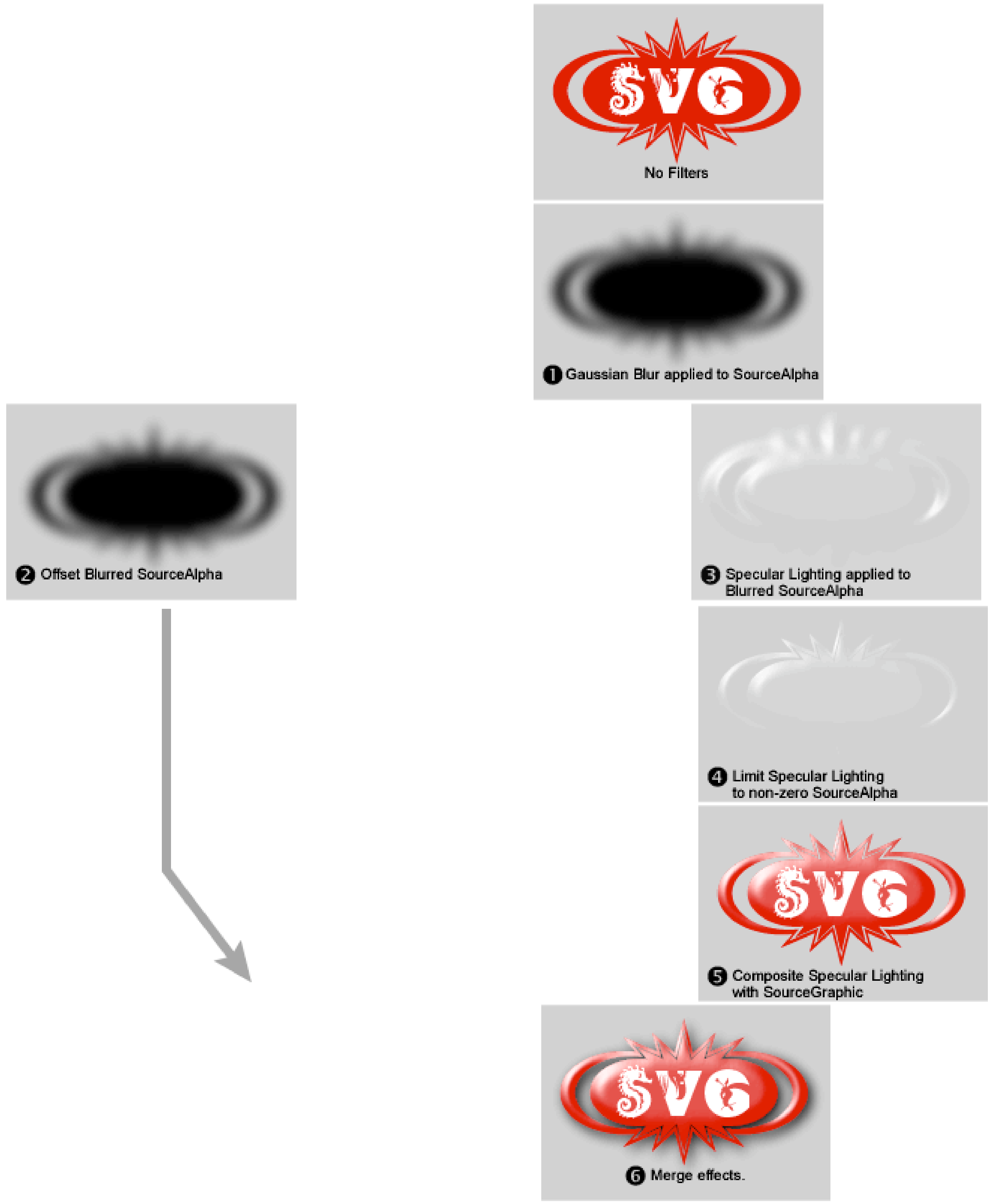
Filter effects

- Two key attributes for filter effects are the SourceGraphic and SourceAlpha.
- The SourceGraphic is the original input for the <filter> element: that is, the graphic element that the filter effect is being applied to.
- The SourceAlpha is the same as the SourceGraphic, except that it contains only the alpha channel. The alpha channel includes any anti-aliasing specified for the element.

Filter effects

The combination of various filters can produce a variety of effects. The image processing model waits until all of the filter effects for an element are compiled (layered) before rendering the final image.

The following image incorporates a combination of Gaussian Blur and Specular Lighting:


No Filters
❶ Gaussian Blur applied to SourceAlpha
❷ Offset Blurred SourceAlpha
❸ Specular Lighting applied to Blurred SourceAlpha
❹ Limit Specular Lighting to non-zero SourceAlpha
❺ Composite Specular Lighting with SourceGraphic
❻ Merge effects.

```
<filter id="MyFilter" filterUnits="userSpaceOnUse">
<!--Copyright 1999 Adobe Systems. You may copy, modify, and distribute this
file, if you include this notice & do not charge for the distribution. This
file is provided "AS-IS" without warranties of any kind, including any implied
warranties.-->
<feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
<feOffset in="blur" dx="4" dy="4" result="offsetBlurredAlpha"/>
<feSpecularLighting in="blur" surfaceScale="5" specularConstant="0.9"
specularExponent="20" lightColor="white" result="specularOut">
<feDistantLight azimuth="135" elevation="30"/>
</feSpecularLighting>
<feComposite in="specularOut" in2="SourceAlpha" operator="in"
result="specularOut"/>
<feComposite in="SourceGraphic" in2="specularOut" operator="arithmetic
" k1="0" k2="1" k3="1" k4="0" result="litPaint"/>
<feMerge>
<feMergeNode in="offsetBlurredAlpha"/>
<feMergeNode in="litPaint"/>
</feMerge>
</filter>
```

Progetti

- comporre visualizzazioni 2D di scene 3D
- usando Javascript per modello e proiezioni
- SVG per output ed effetti
- esperienza interattiva da browser

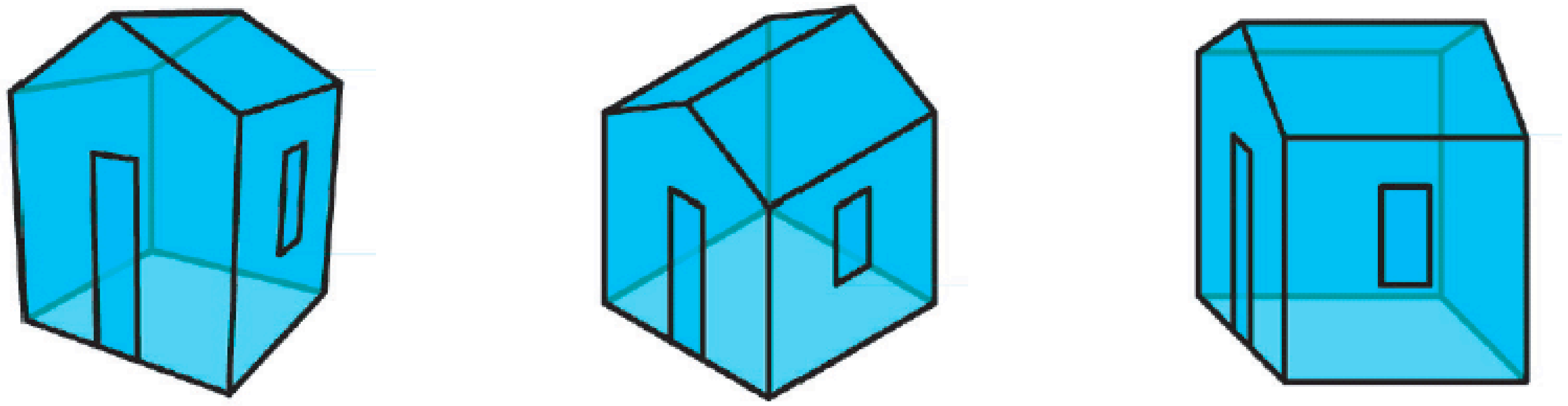


Figure 10.55 Projections exported as Flash files: (a) perspective oblique (b) parallel isometric (c) parallel cabinet

The **house** model is defined in Script 8.5.21; the `FILLCOLOR`, `LINECOLOR`, `LINESIZE` operators, to be used with `FLASH` exporting, are discussed in Section 15.3.2.

Script 10.8.7 (View models)

```
DEF house1 = projection: perspective: threepoints: house;
DEF house2 = projection: parallel: isometric: house;
DEF house3 = projection: parallel: cabinet: house;

DEF out (object::IsPol)(name::IsString) = FLASH:(object
   FILLCOLOR RGBAcolor:<0,1,1,0.5>
   LINECOLOR RGBAcolor:<0,0,0,1>
   LINESIZE 5):300:name;

out: house1: 'house1.swf';
out: house2: 'house2.swf';
out: house3: 'house3.swf';
```