

# Layout Managers for Scalable Vector Graphics

*Keywords:* Layout Managers, Scalable Vector Graphics

## **Ph.D. Martin Rotard**

Researcher in Computer Graphics and Human Computer Interaction  
[University of Stuttgart, Visualization and Interactive Systems Institute](#)  
Universitaetsstr. 30  
Stuttgart  
70569  
Germany  
[rotard\(A.T.\)vis.uni-stuttgart.de](mailto:rotard(A.T.)vis.uni-stuttgart.de)

### *Biography*

Martin Rotard has been staff-member and PhD student of the Visualization and Interactive Systems Institute, University of Stuttgart, since June 2001. His research interests are new graphical standards on the web like Scalable Vector Graphics, MathML, X3D, SMIL, etc., Human Computer Interaction, Usability, and Accessibility. He was engaged in the multimedia-teaching project ITO (Information Technology Online) that was funded by the German Federal Ministry of Education and Research. His development activities are in the field of XML-based graphics, universal access on the web, e-learning, graphical user interfaces and software usability.

## **Prof. Dr. Thomas Ertl**

Professor of Visualization and Computer Graphics  
[University of Stuttgart, Visualization and Interactive Systems Institute](#)  
Universitaetsstr. 30  
Stuttgart  
70569  
Germany  
[ertl\(A.T.\)vis.uni-stuttgart.de](mailto:ertl(A.T.)vis.uni-stuttgart.de)

### *Biography*

Thomas Ertl received a masters degree in computer science from the University of Colorado at Boulder and a PhD in theoretical astrophysics from the University of Tuebingen. Currently, Dr. Ertl is a full professor of computer science at the University of Stuttgart, Germany and the head of the Visualization and Interactive Systems Institute (VIS). Prior to that he was a professor of computer graphics and visualization at the University of Erlangen where he lead the scientific visualization group. Besides that, he is a cofounder and a member of the managing board of science+computing ag, a Tuebingen based IT company. His research interests include visualization, computer graphics and human computer interaction in general with a focus on volume rendering, flow visualization, multiresolution analysis, parallel and hardware accelerated graphics, large datasets and interactive steering. Dr. Ertl is coauthor of more than 150 scientific publications and he served on several program and paper committees as well as a reviewer for numerous conferences and journals in the field.

---

# Abstract

---

In Scalable Vector Graphics (SVG) shapes or groups have to be positioned in absolute coordinates. As an alternative they can be placed relative to the size of the entire canvas. For the author of a SVG graphics it is more comfortable to use a What You See Is What You Get (WYSIWYG) editor, where the placement is implicitly given, instead of calculating each position manually. Another method is writing a script that determines the positions, but this is rather time-consuming. Especially in business graphics or user interface design shapes are placed in a strict horizontal or vertical manner. Layout managers would help the author to position the shapes or groups relative to another element.

In this paper we present layout managers for flow layout, where the shapes are arranged in a horizontal or in a vertical series, and for grid layout, where the area is split into a number of columns. The layout managers can be combined and they are easy to integrate into SVG. The layout is not based on constraints, which makes the configuration and usage simple. In order to determine the position of the shape or group the bounding box has to be calculated. As an outlook we present an approach on how this layout managers could be realized using SVG's XML Binding Language (sXBL).

---

## Table of Contents

---

### [1. Motivation](#)

### [2. Layout Managers](#)

#### [2.1 Flow Layout](#)

#### [2.2 Grid Layout](#)

#### [2.3 Combining the Layout Managers](#)

### [3. Layout Managers using sXBL](#)

### [4. Results and Outlook](#)

### [Acknowledgements](#)

### [Bibliography](#)

## 1. Motivation

Layout managers are used in many toolkits for graphical user interfaces. They implement a policy for deciding how components are positioned within a container. Scalable Vector Graphics (SVG) is an emerging Web standard recommended by the World Wide Web Consortium (W3C) for two-dimensional graphics [\[SVG\]](#). The current version of SVG does not provide a mechanisms for a flexible layout to determine the position of the shapes. In this paper we demonstrate the extension of SVG by layout managers. We present layout managers for horizontal flow layout, for vertical flow layout, and for grid layout.

In SVG shapes or groups have to be positioned in absolute coordinates or they can be placed relative to the size of the entire canvas. The manual calculation of each position is complex and rather time-consuming. Especially in business graphics or user interface dialogs shapes are placed in a strict horizontal or vertical manner. Layout managers can help the authors to position the shapes or groups relative to another element.

In the early phase of the SVG standard constraint-based solutions where already presented [\[Badros et al. 2001\]](#)[\[Marriott et al. 2002\]](#). Recent publications focus on adaptive layout using one-way constraints in SVG to determine the position of the shapes or groups [\[McCormack et al. 2004\]](#). The proposed

constraints are powerful and enable flexible layouts. On the other hand the usage of these constraints is complex and fault-prone. So our proposed layout managers are kept rather simple to configure and easy to use.

The remainder of this paper is structured as follows: [Chapter 2](#) demonstrates the flow layout and grid layout for SVG and shows an example where layout managers are combined. In [Chapter 3](#) we propose how layout managers for SVG can be simplified using SVG's XML Binding Language (sXBL). [Chapter 4](#) draws conclusions and discusses future work.

## 2. Layout Managers

In this section we demonstrate two types of layout managers. The first layout manager determines the layout in a strict horizontal or vertical manner. The second splits the entire canvas into columns. The layout can be applied to group elements and to the "svg"-node. In our demonstrations the "onload"-attribute is used to call the layout managers, which are realized in ECMAScript. For each SVG shape the bounding box has to be calculated. All layouts can be used in combination.

### 2.1 Flow Layout

In flow layout the shapes are arranged in a horizontal or vertical series. Therefore two different commands are defined to call the layout managers:

```
horizontalFlow(evt, spacing)
```

```
verticalFlow(evt, spacing)
```

The first parameter is the standard event parameter to identify the group or "svg"-node, where the event (in this case the "onload"-event) has happened. The second parameter "spacing" defines the spacing between the shapes or groups. The following example code shows a horizontal flow layout in SVG, where the spacing is set to ten pixels. The graphical representation is shown in [Figure 1](#).

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="800" height="400">
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

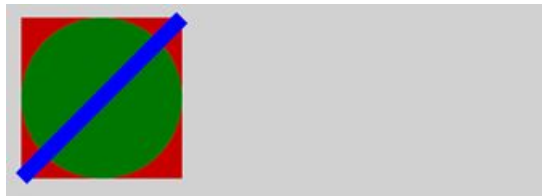
  <g onload="horizontalFlow(evt, 10)">
    <rect x="0" y="0" width="100" height="100" fill="red" />
    <circle cx="50" cy="50" r="50" fill="green" />
    <line x1="0" y1="100" x2="100" y2="0" stroke-width="10" stroke="blue" />
  </g>
</svg>
```



**Figure 1: Shapes in horizontal flow layout**

For each shape in the entire group the bounding box has to be calculated. The dimensions of the bounding box of a group is calculated recursively by considering the size of the elements inside the group element. The layout manager places a shape or group with respect to the position and bounding box of the predecessor group or shape by using the SVG "transform"-attribute. The dimensions of the bounding box of the first shape or group determines the beginning of the second shape or group and so on.

Each shape or group has to be positioned relatively to the origin of the coordinate system. Without using a layout manager the shapes will overlap. [Figure 2](#) shows the rendered document without activating a layout manager.

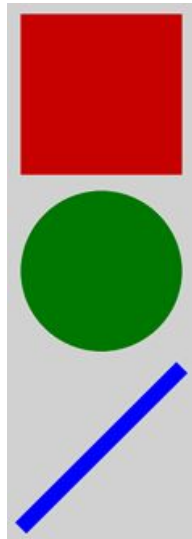


**Figure 2: Positions of the shapes without a layout manager**

The following SVG document shows the example code from above in vertical flow layout. [Figure 3](#) presents the rendering result of the code.

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="800" height="400">
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

  <g onload="verticalFlow(evt, 10)">
    <rect x="0" y="0" width="100" height="100" fill="red" />
    <circle cx="50" cy="50" r="50" fill="green" />
    <line x1="0" y1="100" x2="100" y2="0" stroke-width="10" stroke="blue" />
  </g>
</svg>
```



**Figure 3: Shapes in vertical flow layout**

## 2.2 Grid Layout

In grid layout the entire canvas is split into a number of cells. The number of cells that are horizontally arranged as columns can be configured by a parameter. In contrast to this, the number of rows is not fixed and depends on the height of the cells in each row, which have to be rendered. For each cells in a row the bounding boxes are calculated in the script. The height of the row is determined by the maximum height of the individual cells.

The grid layout manager can be integrated into group and “svg”-nodes by using the "onload"-attribute with the following command and parameters:

```
gridLayout(evt, spacing, columns)
```

The first two parameters for the standard event and the spacing are the same as the parameters of the flow layout. Additionally, the third parameter configures the number of columns that are in a row. In the following example code, a grid layout with two columns and a spacing of ten pixels is applied. The result of the rendering is shown in [Figure 4](#)

```

<?xml version="1.0" encoding="UTF-8" ?>
<svg width="600" height="400">
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

  <g onload="gridLayout(evt, 10, 2)">
    <rect x="0" y="0" width="100" height="100" fill="red" />
    <circle cx="50" cy="50" r="50" fill="green" />
    <line x1="0" y1="100" x2="100" y2="0" stroke-width="10" stroke="blue" />
    <text x="0" y="80" font-family="Verdana" font-size="40" fill="black">SVG Open 2005</text>
  </g>
</svg>

```



**Figure 4: Shapes in grid layout with two columns**

To increase the flexibility of the layout, embedded SVG canvases with grid layout managers can be used. Therefore further “svg”-nodes are included into the document. For these canvases the positions and dimensions can be configured. The grid layout manager determines the positions of all child nodes in the canvas. Especially in the field of graphical user interfaces, the automatic placement of shapes is advantageous. The following example code shows a sketch for a login dialog, where the labels and the input boxes of the forms are positioned by a grid layout manager in an embedded SVG canvas. The graphical representation of the dialog is shown in [Figure 5](#)

```

<?xml version="1.0" encoding="UTF-8" ?>
<svg width="400" height="300">
  <defs>(...)</defs>
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

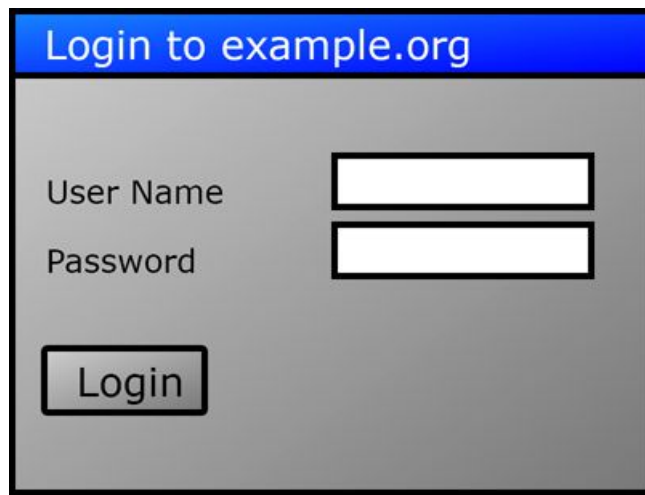
  <rect x="0" y="0" width="400" height="300" class="dialogBorder" />
  <rect x="0" y="0" width="400" height="40" class="dialogTitleBackground" />
  <text x="20" y="50" class="dialogTitle" >Login to example.org</text>

  <svg x="20" y="80" width="360" height="200" onload="gridLayout(evt, 0, 0, 1, 2)">
    <text x="0" y="40" class="dialogLabel">User Name</text>
    <rect x="0" y="10" width="160" height="32" class="dialogForm" />
    <text x="0" y="40" class="dialogLabel">Password</text>
    <rect x="0" y="10" width="160" height="32" class="dialogForm" />
  </svg>

  <rect x="20" y="210" width="100" height="40" rx="2" class="dialogButton" />
  <text x="40" y="240" class="dialogButtonText">Login</text>

</svg>

```



**Figure 5: Sketch for a login dialog, where the forms are positioned in an embedded “svg”-node canvas**

## 2.3 Combining the Layout Managers

The presented layout managers can be combined with each other to achieve complex layout structures. Especially in graphical user interfaces, business charts, and schematic representations this is very useful. In the following example code horizontal and vertical flow layout is combined to build a periodic table of the chemical elements. Each element is represented in a group with a rectangle for the border and two text elements for the chemical element name and the atom number. The structured form of the periodic table allows to calculate the positions once and use them with the flow layout managers for all elements. The first group element has a layout manager for vertical flow layout applied. This determines the vertical placing of the rows. The horizontal layout is achieved by using the horizontal flow layout manager in each row. All chemical elements can be positioned in this way, except the helium element. In the regular layout this element would be placed beside the hydrogen element. Therefore the horizontal coordinates have to be calculated manually. The result of the rendering is shown in [Figure 6](#)

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="800" height="700">
  <defs>(…)</defs>
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

  <g onload="verticalFlow(evt, 1)">
    <g onload="horizontalFlow(evt, 1)">
      <g>
        <rect x="0" y="0" width="100" height="100" class="elemBoxGreen" />
        <text x="50" y="25" class="atomicNr">1</text>
        <text x="50" y="80" class="chemElem">H</text>
      </g>
      <g>
        <rect x="624" y="0" width="100" height="100" class="elemBoxBlue" />
        <text x="674" y="25" class="atomNr">2</text>
        <text x="674" y="80" class="chemElem">He</text>
      </g>
    </g>

    <g onload="horizontalFlow(evt, 1)">
      <g>
        <rect x="0" y="0" width="100" height="100" class="elemBoxRed" />
        <text x="50" y="25" class="atomicNr">3</text>
        <text x="50" y="80" class="chemElem">Li</text>
      </g>
      (….)
      <g>
        <rect x="0" y="0" width="100" height="100" class="elemBoxBlue" />
        <text x="50" y="25" class="atomicNr">10</text>
        <text x="50" y="80" class="chemElem">Ne</text>
      </g>
    </g>
  </g>
```

```

<g onload="horizontalFlow(evt, 1)">
  (...)
</g>
</g>
</svg>

```

1 H							2 He
3 Li	4 Be	5 B	6 C	7 N	8 O	9 F	10 Ne
11 Na	12 Mg	13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
19 K	20 Ca	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
37 Rb	38 Sr	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
55 Cs	56 Ba	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr	88 Ra						

**Figure 6: Periodic table of the chemical elements using layout managers in SVG**

As an alternative layout manager for the periodic table the grid layout with eight columns can be used. The gap between the hydrogen and the helium element can be realized by adding group elements without any child nodes. Another possibility would be to position the hydrogen and the helium element manually and use for the rest of the table the grid layout.

### 3. Layout Managers using sXBL

The handling of the layout managers would be easier by using SVG's XML Binding Language (sXBL), since a language could be realized that defines the tags and attributes for the layout managers [\[sXBL\]](#). These tags could be used in a SVG document, where sXBL defines the mapping of the layout tags. This solution will replace the parameters of the layout managers in comma separated values by a clear and easy to use structure of named attributes. In the following example code the gridLayout is used for the positioning of the shapes. The sXBL-mapping is left out, because the specification of sXBL is in an early phase and has just the W3C working draft state.

```

<?xml version="1.0" encoding="UTF-8" ?>
<svg width="600" height="400" version="1.2"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xbl="http://www.w3.org/2004/xbl"
  xmlns:lm="http://www.example.com/SVGLayoutManagers">
  <script type="text/ecmascript" xlink:href="SVGLayoutManagers.es" />

  <defs>
    <xbl:xbl>
      <xbl:definition element="lm:gridLayout">
        <xbl:template>
          <!-- sXBL-Mapping of the layout manager tag and attributes -->
        </xbl:template>
      </xbl:definition>
    </xbl:xbl>
  </defs>

  <lm:gridLayout spacing="30" columns="2" >
    <rect x="0" y="0" width="100" height="100" fill="red" />
    <circle cx="50" cy="50" r="50" fill="green" />
    <line x1="0" y1="100" x2="100" y2="0" stroke-width="10" stroke="blue" />
    <text x="0" y="80" font-family="Verdana" font-size="40" fill="black">SVG Open 2005</>
  </lm:gridLayout>
</svg>

```

Elliotte Rusty Harold has shown a mapping example of elements in a chemical language into a periodic table in SVG by using sXBL [\[Harold 2005\]](#). In this example the positioning for the chemical elements is missing and could be supplemented with the presented layout managers. The extended version of the periodic table for chemical elements is shown in the following code.

```

<?xml version="1.0" encoding="UTF-8" ?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.2"
  xmlns:chem="http://ns.cafeconleche.org/chemistry/"
  xmlns:xbl="http://www.w3.org/2004/xbl"
  width="101" height="101">

  <defs>
    <xbl:xbl id="atoms" >
      <xbl:definition element="chem:ATOM">
        <xbl:template>
          <rect x="0" y="0" width="100" height="100"
            stroke="black" stroke-width="2px" fill="none"/>
        </xbl:template>
      </xbl:definition>
      (...)
    </xbl:xbl>
  </defs>
  <lm:verticalFlow spacing="1" >
    (...)
    <lm:horizontalFlow spacing="1" >
      (...)
      <chem:ATOM>
        <chem:NAME>Aluminum</chem:NAME>
        <chem:ATOMIC_WEIGHT>26.98154</chem:ATOMIC_WEIGHT>
        <chem:ATOMIC_NUMBER>13</chem:ATOMIC_NUMBER>
        <chem:SYMBOL>Al</chem:SYMBOL>
        <chem:ELECTRON_CONFIGURATION>[Ne] 3s2 p1</chem:ELECTRON_CONFIGURATION>
      </chem:ATOM>
      (...)
    </lm:horizontalFlow>
    (...)
  </lm:verticalFlow>
</svg>

```

## 4. Results and Outlook

In this paper we have proposed layout managers for Scalable Vector Graphics, which are easy to use and configure. We have demonstrated two types of layout managers, one for horizontal and vertical flow

layout and a second for grid layout. The layout managers can be combined to achieve a complex layout structure. The handling could be improved by using SVG's XML Binding Language.

The presented layout managers are still work in progress and should be extended with many functionalities. Especially the number of parameters needs to be increased to configure the horizontal and vertical spacing of the shapes. Some simple constraints could be used to position the shapes horizontally and vertically centered or on the top, bottom, left, or right border of the entire canvas. Therefore embedded SVG canvases can be used.

There are some layout managers that are often used in toolkits for graphical user interfaces like border layout, gridBag layout, overlay layout, etc., which could be realized for SVG. Additional layout managers could be defined to position shapes or groups inside the geometric area or on top of another SVG shape.

## Acknowledgements

We like to thank Cameron McCormack for his inspiration and encouragement to write this paper.

## Bibliography

### [Badros et al. 2001]

Badros, G; Tirtowidjojo, J.; Marriott, K.; Meyer, B.; Portnoy, W.; Borning, A.: *A constraint extension to scalable vector graphics*, Proceedings of the 10th World Wide Web Conference, 2001

### [Harold 2005]

Elliotte Rusty Harold: *An early look at sXBL - When XSLT isn't enough*, IBM developerWorks, 2005, <http://www-128.ibm.com/developerworks/xml/library/x-sxb11/>

### [Marriott et al. 2002]

Marriott, K.; Meyer, B.; Tardif, L.: *Fast and Efficient Client-side Adaptivity for SVG*, Proceedings of the 11th International World Wide Web Conference, 2002

### [McCormack et al. 2004]

McCormack, C. ; Marriott, K.; Meyer, B.: *Adaptive layout using one-way constraints in SVG*, Proceedings of the SVG Open Conference, 2004  
<http://www.svgopen.org/2004/papers/ConstraintSVG/>

### [SVG]

World Wide Web Consortium: *Scalable Vector Graphics (SVG)*, <http://www.w3.org/TR/SVG/>

### [sXBL]

World Wide Web Consortium: *SVG's XML Binding Language (sXBL)*,  
<http://www.w3.org/TR/sXBL/>