

---

# **S60 Platform: Vector Graphics Optimization**

**Version 1.0**  
February 23, 2006

**S60** platform

## Legal Notice

Copyright © 2006 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

### Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

### License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

## Contents

<b>1.</b>	<b>Introduction .....</b>	<b>5</b>
<b>2.</b>	<b>File Formats .....</b>	<b>5</b>
<b>3.</b>	<b>Optimizing Graphics .....</b>	<b>6</b>
3.1	File Size.....	6
3.2	Size in Pixels .....	6
3.3	Complexity.....	7
3.4	Reducing Points .....	7
3.4.1	Simplify curved points.....	8
3.4.2	Simplify shapes .....	9
3.5	Demanding Features.....	10
<b>4.</b>	<b>Designing Vector Graphics .....</b>	<b>11</b>
4.1	Most Common SVG Tools .....	11
4.2	How to Get Easily Started .....	12
4.3	When to Use Vector Graphics .....	12
4.4	About SVG to SVG-T Converter .....	13
<b>5.</b>	<b>References .....</b>	<b>14</b>
<b>6.</b>	<b>Evaluate This Resource.....</b>	<b>14</b>

## Change History

February 23, 2006	Version 1.0	Initial document release

## 1. Introduction

This document gives instructions for optimizing vector graphics for the S60 3rd Edition user interface. The purpose of these instructions is to:

- Increase rendering performance.
- Minimize memory consumption.
- Enlighten some points of designing/using scalable vector graphics in mobile device environment.

## 2. File Formats

The S60 3rd Edition SVG rasterizing engine supports the SVG Tiny (SVGT) 1.1 standard with additional support for

- Line and fill opacity.
- Gradient.

File format	SVG Tiny
Compression	No (ASCII text or Binary)
Color depth	24 bits / pixel
Masking files	No (implemented with opacity values from SVG file )

**Table 1: Vector graphics type**

SVG graphics can be converted to the SVGT format using the SVG To SVGT Converter tool delivered with the *S60 3rd Edition SDK*.



**Note:** The screen module or SVG engine rendering color depth may be less than 24 bits / pixel. The graphics are shown 'as is' without using any image dithering methods — this may result in banding in large gradient areas.

The SVG file format exported from a standard vector graphic authoring tool is supported. Due to different implementations of the SVG standard, file format compatibility problems may occur.

Graphics containing other features than what have been defined in the SVG Tiny standard [1] or supported by the S60 3rd Edition SVG rasterizing engine may not be displayed correctly.

Vector graphics are rendered with a smooth anti-aliasing effect on borders. This produces a small gap between two adjoining vectors even if the edges are numerically matched without any gap. This can be avoided by overlaying the edges.

## 3. Optimizing Graphics

Graphics optimization can be evaluated with different criteria:

- File size.
- Size in pixels.
- Complexity.
- Features required.

### 3.1 File Size

Large file sizes increase the memory consumption and correspond generally well with slow rendering performance.

UI graphics need to be checked for large file sizes. More complex and detailed designs as well as large graphics consume more memory.

If a file size is bigger than 10 KB, the file contents need to be re-evaluated to simplify the design (see Section 3.3, “Complexity”).

Template	Size (KB)*	Remarks
File header	0.323	SVG2SVGT tool output
Average shape	0.1~1	Rectangular path with 10 curved anchor points
Gradient	0.26	3-color, one more color += 0.085

\*: SVGT file's storage size on disk, ASCII text format optimized with SVG2SVGT tool

**Table 2: SVGT file template sizes**

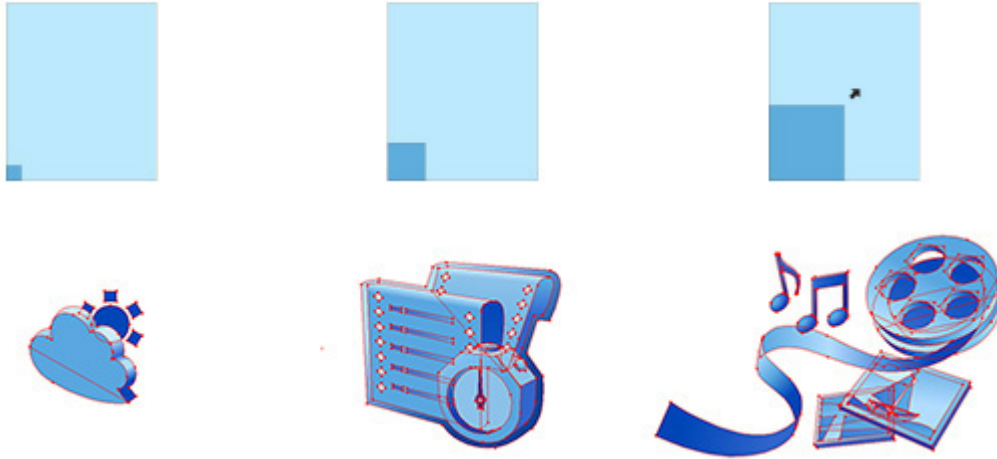
### 3.2 Size in Pixels

SVGT graphics are rendered to the defined layout area. The area aspect ratio and size may be different from the original graphics — the graphics are scaled and stretched to fill the area.

The required area sizes affect the graphics design: the larger the screen size, the more complex and detailed the design can be — with a smaller screen size less details and a simpler design are preferred. Images can be divided into three size classes:

- Small (10% of screen width)                      simple design
- Medium (25% of screen width)                    more complex design possible
- Large (50% to full screen)                        most complex design possible

Example:



The target sizes for the different graphics types are:

Graphics group	Size	Remarks
Menu graphics	<10 KB	(exceptions possible)
Note graphics	<7 KB	
Submenu graphic	<7 KB	
Large list graphics	<7 KB	
Medium list graphics	<7 KB	
Small list graphics	<4 KB	

**Table 3: Target sizes**

### 3.3 Complexity

The complexity of the graphics is mainly defined by the content and drawing style. A world map or a gothic-looking object is relatively bigger in file size than a ball or a simplified modern-looking object.

Complexity can be reduced without affecting the outlook of the object by:

- Reducing points (anchors)
- Reducing curved points
- Simplifying shapes (paths)

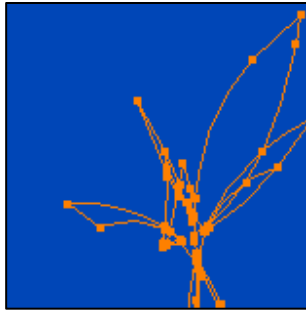
### 3.4 Reducing Points

Removing unnecessary points is a way to optimize graphics because each point consumes ~20-40 bytes of memory. Actions to do this are:

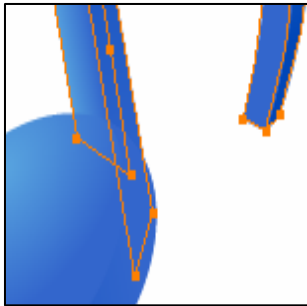
- Removing stacked points.
- Removing points that have no visual impact.

Points may be created stacked close to each. This may occur, for example, when multiple shapes have been combined with Boolean operations (for example, union, subtract, divide).

Example:



Remove points that do not have any visual effect — a point's existence does not affect the shape's outlook.



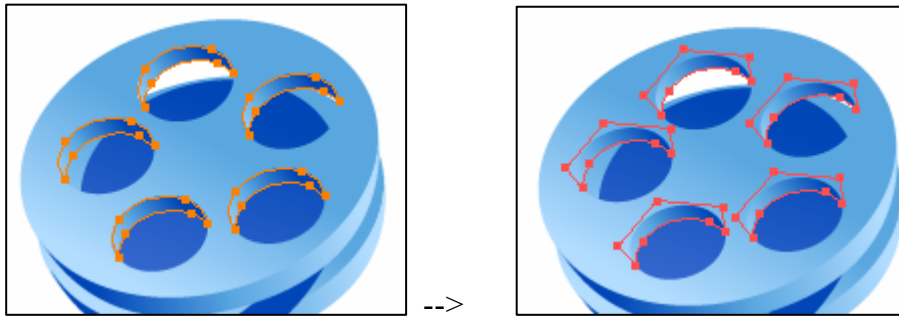
### 3.4.1 Simplify curved points

A curved line requires more time and CPU power to render than a straight line. Therefore utilizing straight lines whenever possible is preferable. Actions to do this are:

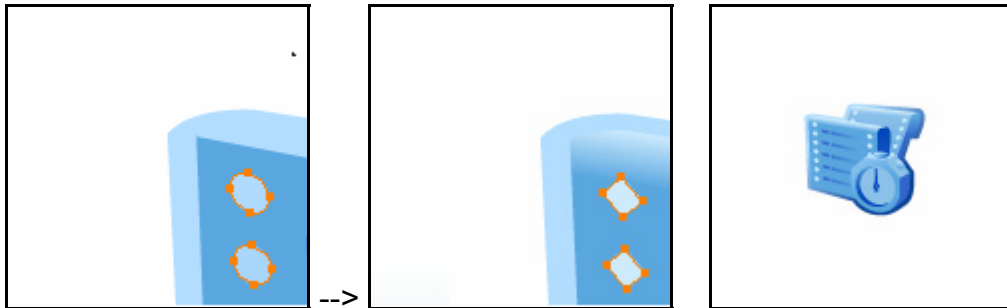
- Convert hidden curved lines to straight lines.
- Change visually indifferent curved lines to straight lines.
- Retract unnecessary handles.

If an outline is hidden, it is recommended to simplify it by converting the relevant curved points to straight ones.

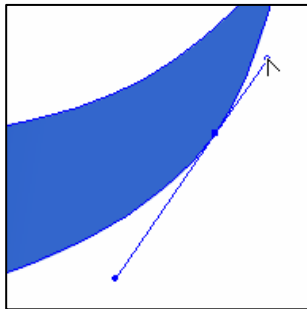
Example:



If curved lines cannot be recognized, convert them into straight lines.



If a line does not need to be curved on both sides of a point, retract the handle that is on the straight line side.

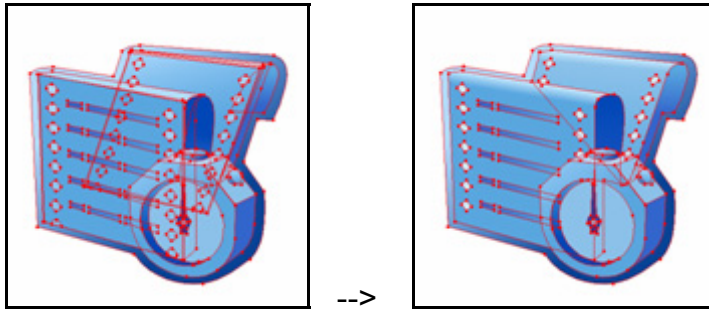


### 3.4.2 Simplify shapes

Simplifying shapes by removing unnecessary details is another step for optimizing graphics. Actions are:

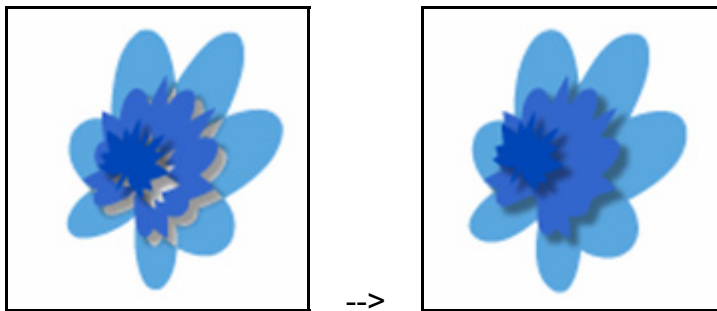
- Remove hidden shape details.
- Combine shapes with similar fills.
- Remove hidden graphics details.

Example:



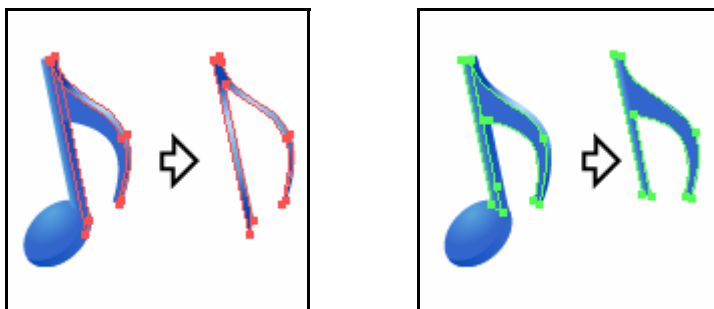
However, when graphics are overlapping, do not subtract foreground shapes from the graphics behind them because it creates overlapping edges and increases point count.

Example:



Combine connected shapes that have the same or similar fill.

Example:



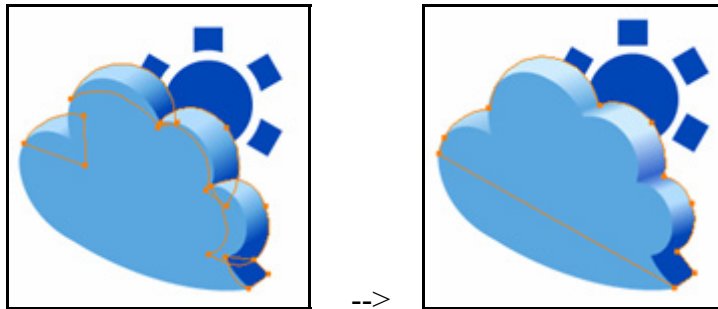
### 3.5 Demanding Features

Gradient and opacity are relatively demanding features for the SVG engine to render, especially when used in large areas. Optimization actions are:

- Use gradients and opacity sparingly.
- Combine shapes with the same gradient.
- Do not use overlapping transparency (opacity).

Unify shapes that can share the same gradient.

Example:



Avoid using overlapping transparency or transparency on a gradient. Create the same visual effect 'faking' the transparency outcome.

## 4. Designing Vector Graphics

### 4.1 Most Common SVG Tools

This section lists some examples of the most common tools capable of saving vector graphics drawings in SVG format, either directly or indirectly. Also some usage experiences are given.

Professional tools	
Adobe Macromedia (Freehand)	<ul style="list-style-type: none"> <li>• Very agile in drawing images.</li> <li>• Cannot save graphics directly in SVG format.</li> <li>• Available for PC and Mac.</li> <li>• Application is not developed further.</li> </ul>
Adobe Illustrator	<ul style="list-style-type: none"> <li>• Not so easy or flexible to use when drawing graphics.</li> <li>• More flexible in layout setting and other issues.</li> <li>• Can save graphics directly in SVG format.</li> <li>• Available for PC and Mac.</li> </ul>
Other tools	
Inkscape <a href="http://www.inkscape.org">www.inkscape.org</a>	<ul style="list-style-type: none"> <li>• Easy to use and free.</li> <li>• Can save graphics directly in SVG format.</li> <li>• Made for SVG editing.</li> </ul>

<p>Gimp <a href="http://www.gimp.org">www.gimp.org</a></p>	<ul style="list-style-type: none"> <li>• Can save graphics directly in SVG format.</li> <li>• Not only an SVG editor.</li> <li>• Works in many environments (e.g. PC, Linux, and Mac)</li> </ul>
<p>Sketsa <a href="http://www.kiyut.com/products/sketsa">www.kiyut.com/products/sketsa</a></p>	<ul style="list-style-type: none"> <li>• Can save graphics directly in SVG format.</li> <li>• Made for SVG editing.</li> <li>• More complex than, for example, Inkscape.</li> </ul>

Basically all graphics editors may be used in SVG creation, but since they all do not offer the possibility to save directly in SVG format, the graphics must be somehow converted to SVG. For example, when using Macromedia Freehand, graphics have to be saved first in EPS or Illustrator (ai) format and then saved with Illustrator in SVG format. Only after this, the graphics can be converted to SVG-T.

Currently the biggest shortcoming with any of the above-mentioned editors is that they do not offer the possibility to create SVG-T files directly. A separate converter is needed to convert the SVG files to SVG-T.

Another problem when using graphics editors in creating XML-based graphics is that the editors create excessive code lines when compared to writing the code manually. On the other hand, “manual drawing” is more suitable for simpler icons and images, such as simple circles. For example, the Inkscape editor draws a circle using 5 code lines, whereas it may be manually coded using only one line. This is because the editor does not draw the circle using the circle XML tag, but makes it a multipoint graphical element. And this, again, is because it is possible to modify the circle to an ellipse, and so on. In general, manually-coded graphics are not favored due to their simple outlooks.

## 4.2 How to Get Easily Started

One way to get easily started is to install a free SVG editor. The free editors may be used in evaluation and in general when learning to use SVG graphics. Later on a more professional tool may need to be used.

After you have installed the application, just open it and start drawing. Once the drawing is ready (keep it simple to avoid complications when doing it for the first time), it may be saved in the proper file format. When saving, the Plain SVG option must be selected.

The created SVG file must be converted to the SVG-T file by using the Svg2SVGt tool (see Section 4.4, “About SVG to SVG-T Converter” for more information).

## 4.3 When to Use Vector Graphics

Vector graphics should be used whenever a view element is the same in different resolution screens or it is more complex than a simple set of lines. This enables the same application code to be used in different devices without having, for example, multiple icons for different resolutions. A simple vector graphics icon looks the same inspite of the used resolution. Another advantage is that there is no need for separate mask files in the platform.

Vector graphics are supported from S60 2nd Edition, Feature Pack 3 onwards. Vector graphics animation is not supported by the S60 platform. The feeling of animation can be obtained by sequentially displaying separate images.

#### **4.4 About SVG to SVG-T Converter**

The SVG to SVG-T converter is a tool that converts SVG files to SVG-T files. It is included in the S60 SDK.

However, the tool has some limitations, or more accurately, the SVG file that is about to be converted may only have a limited set of features. By keeping the created SVG files simple, the conversions should be successful.

The converted SVG-T files' appearance should always be verified, for example, by opening them in a Web browser to guarantee that their outlook is as desired.

---

## 5. References

- [1] Scalable Vector Graphics (SVG) 1.1 Specification, available at <http://www.w3.org/TR/SVG11/>

Further reading:

[S60 Platform: Scalable Screen-Drawing How-To](#), available at [www.forum.nokia.com](http://www.forum.nokia.com)

---

## 6. Evaluate This Resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by [rating this resource](#).