# MS RFC – SVG symbol Support with AGG

Kiran Anjaneya Varma Alluri

August 10, 2009

## 1 Overview

Popular tools such as Inkscape generate SVG graphics natively. So far MapServer has not supported the SVG format, which leaves many users with the extra work of converting between formats. This RFC aims to introduce support for SVG graphics as an alternate method to define vector symbols in the MapServer symbolset.

## 2 Method

The SVG parsing and rendering will be triggered by the following syntax:

```
SYMBOL
    TYPE SVG
    SVGPATH /path/to/file.svg
END
```

The choice for using a SYMBOL for the task is to allow for the reuse of the SVG symbol easily within classes.

## 3 Technical Overview

In order to support the SVG format, we make use of the AGG library. The reasons for going with the AGG library are:

1. It has already been integrated into MapServer. While other popular libraries, such as Cairo, can be used, they would require effort to integrate into MapServer and would further introduce many dependencies into the MapServer code.
2. It is a very lightweight library and delivers performance equal to, and sometimes higher than other popular libraries.

There are some disadvantages that emerge from using the AGG library. Foremost, the AGG library has an incomplete SVG specification. To avoid having to reimplement the missing subset of the specification (which would, without a doubt, be a non-trivial task, and would add to the complexity of this RFC), we stick by the following techniques:

1. A modular design is used, where the generation of the intermediate pixmap is done by a single function so that it can be easily modified to adjust to a diferent library. Thus, using a pixmap will allow us to use any rendering library.

2. The unimplemented subset of the SVG specification is clearly provided to users in the documentation to prevent frustration on their side.

To implement this functionality, the following steps needs to be accomplished:

1. Implement an SVG parser. For this, we directly make use of the SVG Viewer example provided along with the AGG sources.
2. Modify the MapServer lexical analyzer and parser to recognize the new keywords (SVG and SVGPATH).
3. Modify the symbolObj structure to store information about the SVG symbol requested (eg. File path).
4. Write a single function to render a parsed SVG file to a pixmap. The pixmap will be an intermediate form that can then be rendered. The pixmap will be stored in cache memory to prevent having to constantly be rasterized it every time a symbol is drawn.
5. Create a createBrushFromSVG() function that creates a GD image pixmap. Also modify the basic rendering functions to support the new symbol type by using the createBrushFromSVG function.
6. Render the pixmap into the final image file. This will be done using a mixture of the AGG and GD libraries.

# 4 Files Added and Modified

## 4.1 AGG/SVG Parser

The SVG parsing and pixmap generation functions will be stored in source files stored at

/ mapserver / renderers / agg / svg /

The files within the above directory include the files for:

- SVG Parsing
- Trigger Function (based on Mapfile syntax)
- SVG to Pixmap generation

All files are within the C++ 'mapserver' namespace.

## 4.2 Rendering Functions

The rendering functions in mapgd.c will be modified to support the SVG type. These include all the basic shapes such as points, lines, circles etc. These will make use of the createBrushFromSVG function to render to a GD pixmap.

## 4.3 Lexical Analyzer and Parser

The maplexer.l file will be modified to support two new keywords (SVG and SVGPATH). The mapsymbol.h file be modified by adding SVG related data to the symbolObj structure along with basic enumerations and defines. The mapsymbol.c file's loadSymbol function will be modified to generate the required new fields in the symbolObj structure.

# 5 Documentation

The following documentation needs to be prepared for this project:

1. Missing SVG specification document.
2. Basic usage tutorial of new functionality with examples.