Generating SVG weather maps and meteorological graphs using Magics++

Stephan Siemen
Graphics Section
ECMWF
Outline

- What is ECMWF?
- What is Magics++?
- How did we implement our SVG output?
- Some examples
- Can we use SVG for our web products?
What is ECMWF?

- European Centre for Medium Range Weather Forecasts

- We provide operational medium- and extended-range forecasts and a state-of-the-art super-computing facility for scientific research.

- Supported by 31 States

- 220 Employees

- Founded 33 years ago

- Based in Reading, west of London, United Kingdom.
Supporting States and Co-operation

Belgium  Ireland  Portugal
Denmark  Italy  Switzerland
Germany  Luxembourg  Finland
Spain  The Netherlands  Sweden
France  Norway  Turkey
Greece  Austria  United Kingdom

Co-operation agreements or working arrangements with:

Czech Republic  Montenegro  ACMAD
Croatia  Morocco  ESA
Estonia  Romania  EUMETSAT
Hungary  Serbia  WMO
Iceland  Slovakia  JRC
Latvia  Slovenia  CTBTO
Lithuania

ECMWF
What is Magics++?

- Graphics library to visualise meteorological data
- Meteorological- and object-oriented design; flexible to cope with model changes and new types of data
- Outputs are high quality technical maps → for forecasters, decision makers and researchers
- Various APIs: Fortran, C++, MagML (XML)
- Scripting interfaces such as Python are added using SWIG
- Freely available under Apache license
- Tested on various UNIX derivates (Linux, AIX, MacOSX)
Magics++ black box

Programming interfaces

Data-Input

GRIB 1 & 2
BUFR
NetCDF
ODB
Matrices
MapGen
Geopoints

New contouring

Features new in Magics++

Output

PostScript & EPS
PDF
PNG & GIF
SVG
KML
OpenGL

MagML

Plot on demand

C++
C
SWIG
Python
Fortran
What can Magics++ do?

- High quality contouring
  - Supports various levels of quality thanks to AKIMA
- Automatic and user specified titles
- Automatic legends
- WMO observation plotting (BUFR)
- Visualisation of feedback data (ODB)
- Display of satellite data
- Supports geographical and cartesian projections
- Allows users to control layout of plots on pages
Magics++ - MagML

- XML based format to describe Magics++ plots
- Descriptive, not procedural
- Some support of variables and global definitions
- No need for (re-)compilation

magml interpreter program processes a MagML file and produces output graphics

Interpreter can be easily called in user code

Can be integrated into more complex XML request descriptions
MagML - example - the code

<magics version="3.0">
  <drivers>
    <svg name="mysvg.svg" />
    <png name="mypng.png" />
  </drivers>
  <page orientation='landscape'>
    <text font='Times-Roman' colour='rgb(0.1,0.6,0.2)' bottom='90%' display='inline'>Z500 Contour Plot</text>
    <map>
      <cylindrical>
        <corners min_latitude='25' max_latitude='75'
          min_longitude='-30' max_longitude='77'/>
      </cylindrical>
      <coastlines>
        <coast colour = 'brown'/>
      </coastlines>
      <layer>
        <grib path='null'>${PATH_TO_DATA}'
        <contour/>
      </layer>
    </map>
  </page>
</magics>

Variables can be passed to interpreter
MagML - example - the output
Magics++ - next steps

- Rotated grids
- Observation plotting
- Support for BUFR metgram data
- Python interface
- “Style sheets” for data types / products
- Investigate streamlines
- Introduce new projections (e.g. tephigrams)
- Investigate advanced GIS features (e.g. shaded relief maps)
Advantages of using SVG

- Vector format
  - scalable
- Web based
- Interactivity is supported
  - JavaScript well supported by browsers
- Human readable
  - Speeds up development
- Possible print format?
  - SVG Print?
Implementation (1)

We decided to have two approaches:

- Based on the Cairo graphics library
- Own development to generate SVG code

Both implementations turned out to be mostly straightforward

- Exception were raster data (satellite and overlays)

Even though the Cairo based driver turned out well we decided to go with our own driver, because

- No scripting support in SVG generated in Cairo
- Large size of raster output
- Graphic itself was not scalable (hard coded size)
- Different versions of library and most clients have old one
Implementation (2)

- We used GD graphics library to generate externally linked raster files
  - Needed for satellite
  - External files are not nice to handle

- Positioning of Text (especially contour labels) was a challenge
  - Not all attributes for positioning are supported by Firefox
Implementation - summary

- Very easy implementation
- Standard gives us all that we need
  - but not all is supported by viewers!
- External files are hard to handle in an automatic system

- Overall: SVG is one of the easiest and fastest to implement formats we have supported
Scalability

Without fixed size an SVG graphic adjusts its size to the viewer.
Interactivity (1)

A magnifier glass was implemented using JavaScript.
Interactivity (2)

Layers of information can be defined and toggled on and off. Implemented using JavaScript.
Using SVG for our web products

- Most maps (more than 200,000!) are generated daily for ECMWF’s web page for forecasters around the world.
- Currently we generate PostScript which gets converted to GIF and PDF.
- SVG in theory could replace all 3 if used.
- Internet Explorer support is a crucial topic since 80% users use Internet Explorer.
- External dependent files are not easy to handle.

- We investigate now OGC web services
  - SVG might play a bigger role there.
Conclusion

- SVG was easy to implement
- Users who have seen SVG are excited with quality and interactive features
- Drawbacks: IE support & SVG Print

→ We will actively support SVG in future but it will not be used operationally yet

- For more information about Magics++ see
  http://www.ecmwf.int/publications/manuals/magics/magplus