

The gridSVG package

Paul Murrell

February 12, 2004

Introduction

This package is an experiment in writing a graphics device, purely in R code, for the grid graphics system. The specific device implemented is for the W3C SVG (Scalable Vector Graphics) format, but there is also some effort at a general device interface that would support other specific devices.

User Interface

There are three functions of interest to the user:

`grid.hyperlink` takes a grid `grob` and turns it into an object of class `linked.grob`, with an associated `href`. This allows the association of hyperlinks with elements of a grid graphic. See `gridSVG/tests/testlink.R` for examples.

`grid.animate` allows the user to associate a duration (plus some other things) with a grid `unit`. This allows a grid graphic element to be animated. See `gridSVG/tests/testanimate.R` `testpendulum.R` and `testball.R` for examples.

`gridToSVG()` saves the current grid graphic to an SVG file. See the `gridSVG/tests` directory for examples of what can be done. See the section “Known Problems” below for things that are not yet supported.

In addition to these functions, `gridSVG` supports alpha-transparency by respecting the `alpha` graphical parameter which can be specified in a `grid gpar` object. For example, the following code produces overlapping transparent circles¹:

```
> pushViewport(viewport(gp = gpar(col = "black", fill = NA)))
> grid.circle(x = 0.33, r = unit(2, "inches"), gp = gpar(alpha = 0.3,
+   fill = "red"))
> grid.circle(x = 0.67, r = unit(2, "inches"), gp = gpar(alpha = 0.3,
```

¹The `pushViewport()` call is currently necessary to set some default values. It may be possible to remove this in future versions.

```
+      fill = "green"))  
> popViewport()  
> gridToSVG()
```

Internal Structure

There are seven .R files in the `gridSVG/R` directory, corresponding to the seven different things that gridSVG aims to provide:

dev.R This contain (S4 methods) code defining a generic R-level graphics device interface. In other words, generic functions that may be called by a graphics system (such as grid), and that a graphics device (such as an SVG device) should provide methods for.

griddev.R Code for running through the grid display list and calling generic device functions.

devsvg.R Code implementing SVG methods for the generic device interface.

svg.R A set of R-level functions for producing SVG output. Callable directly (see, e.g., `gridSVG/tests/testsvg.R`), but mostly just called by code in `devsvg.R`.

gridsvg.R The function `gridToSVG()`.

hyper.R Code implementing the `linked.grob` class – i.e., an extension of the standard grid `grob` that supports hyperlinks. Includes the function `grid.hyperlink()`.

animate.R Code implementing the `time.unit` class – i.e., an extension of the standard grid `unit` that supports animation. Includes the function `grid.animate()`.

Known Problems

This package is a partial implementation of several ideas. This section describes some of the known holes in the implementation.

Overall Design

The package is ass-backwards in its design. Normal devices receive calls from grid to perform operations; gridSVG works off grid's display list so only has the information stored there to figure out what to do. This means that it has to replicate some of the work that grid does when grid draws (e.g., in order to enforce vp slots in grobs). If/when normal devices are implemented as R-level objects, so that grid includes a `dev` argument in all its calls to devices, it may be possible to make gridSVG behave more like a normal device and this may lead to some simplifications.

Plotting Symbols

Only `pch=1` and `pch=3` are currently supported. This is just a matter of filling in the other options.

Mathematical Annotation

The use of things like `grid.text(expression(x[i]))` is supported in the main R graphics engine. `gridSVG` bypasses that and so does not support mathematical annotation (and probably never will!).

Time unit arithmetic

Arithmetic with time units is not yet supported. I had a go at it, and I think the basic idea is sound, but I think the problem is that S3 method dispatch cannot handle it (see `WAS.Ops.time.unit` in `animate.R`). This will have to wait until `grid` has been reimplemented using S4 methods.

This is a fairly serious limitation as it means that the result of any arithmetic involving a time unit will NOT be a time unit. This means that the current simple implementation of grid points and non-left-bottom-justified rectangles (for example), cannot be animated.

NOTE that this problem does not involve something that the user will necessarily want to do, but the `gridSVG` code that converts the display list into SVG will certainly try to do it.

One way out of this problem may be to use the `grid.convert` function to “evaluate” a complex unit arithmetic expression into a simple unit. The device size does not change when the display list is converted into an SVG file so this should work properly.