Drawing Contexts in Grid Graphics

Paul Murrell

July 9, 2003

In Grid, all drawing is ultimately performed by a set of predefined primitives (lines, rectangles, text, ...) and the output produced by a primitive depends on two sorts of things:

1. the location and size of the primitive; I call these *structural* parameters.
2. the colour, linewidth, fontsize, ... of the primitive; I call these *graphical* parameters.

Furthermore, all drawing occurs within a viewport, which consists of three sorts of things:

1. the location and size of the viewport (structural parameters).
2. the colour, linewidth, fontsize, ... of the viewport (graphical parameters).
3. a number of coordinate systems, which I call the *drawing context* defined by the viewport.

For both primitives and viewports, the same set of parameters may produce different final results when drawn within different drawing contexts. One consequence of this is that, in order to replicate the result from drawing a primitive or setting a viewport (e.g., when redrawing an image), it is vital that the same drawing context can be reestablished.

1 Structural parameters

Location and size parameters are specified using `"unit"` objects. For example, `unit(1, "inches")` specifies a location or size of 1".

Some units are absolute (such as inches and centimetres), but most are relative to the current drawing context - the coordinate systems defined by the current viewport.

In this sense, the location and size of objects are *declarative*.

2 Graphical parameters

Graphical parameters are specified using a `"gpar"` object. A primitive or viewport does not have to specify any or all possible graphical parameters. Anything that is specified overrides any previous (parent) settings and provides the default for any subsequent (child) settings.

In this sense, graphical parameters are *inherited*.

A primitive's graphical parameter settings take effect just before the primitive is drawn, and their effects are reversed just after the primitive has finished drawing[1].

A viewport's graphical parameter settings take effect just before the viewport is pushed onto the viewport stack, and their effects are reversed just after the viewport is popped off the stack.

3 The Drawing Process

Because primitives and viewports contain a declarative specification of their location and size, and because graphical parameters are inherited, a lot of calculation is required *at drawing time* to determine where the output should end up and what it should look like.

For example, the following command requires establishing the current physical location and size of the parent viewport in order to determine the location that is `1"` in from the bottom-left corner, and the current fontsize must be established in order to determine how large the text should be.

```
grid.text("hi", x=1, y=1, default.unit="inches")
```

4 Coordinate Systems

Each coordinate system within a viewport is specified through one or more parameters. For example, the `"native"` units on the x-dimension are specified through a minimum value and a maximum value.

Most of these coordinate system parameters are stored in the viewport, however, two of the coordinate systems have special parameters. The `"lines"` and `"char"` units are based on the current `lineheight` and `fontsize`, which are general graphical parameters. Viewports *can* have a lineheight and/or fontsize specified, but this is not required. For example, a viewport may not specify either of fontsize or lineheight if it just wants to inherit these values from its parent viewport.

[1] Technically, the graphical parameters are only in effect during the call to the `draw.details` method

The drawing process requires that we can establish things like the current font-size and lineheight for each viewport.

5 Addendum

(I can now see that) The separation into structural and graphical parameters mirrors the HTML/XML/SVG separation of structure and appearance (using e.g., CSS). Can't see yet what coordinate systems correspond to in this analogy, but things may become clearer with more reading about SVG.