The Relationship Between Parents and Children in Grid Graphics

December 17, 2001

In Grid there are many ways to specify the location of an object. The most simple is to use one of the "basic" units. For example,

⟨*basic*⟩≡
```
grid.rect(x=unit(1, "inches"), y=unit(.1, "npc"),
          width=unit(.5, "native"), height=unit(1, "lines"),
          just=c("left", "bottom"))
```

The idea here is that the child knows exactly where it wants to go within its parent (viewport).

The other "basic" alternative is to use layouts. For example,

⟨*basic*⟩+≡
```
grid.newpage()
push.viewport(viewport(layout=grid.layout(2, 2)))
push.viewport(viewport(layout.pos.col=2, layout.pos.row=2))
grid.rect()
pop.viewport(2)
```

The idea here is that the parent knows exactly where it wants the child to go.

More complex situations are possible. These are accessed via `"grobwidth"` and `"grobheight"` units, which are the basis of the `grid.frame` and `grid.pack` functions. For example,

⟨*complex*⟩≡
```
grid.newpage()
gt <- grid.text("howdy")
grid.rect(width=unit(1, "grobwidth", gt), height=unit(1, "lines"))

grid.newpage()
gf <- grid.frame()
grid.pack(gf, grid.rect(x=unit(1, "npc"),
                        w=unit(1, "char"), h=unit(1, "char"),
                        just=c("right", "centre"), draw=F))
grid.pack(gf, grid.text("A label", draw=F), side="right")
```

The idea here is that the parent asks the child how big it should be and allocates the required space, but takes care of the locations of its children.

When we are dealing with this sort of complex situation, we have to start thinking about what sort of unit is being used. In particular, we make the distinction between *absolute* units and *relative* units. The clearest examples of absolute units are the physical units such as inches and centimetres. In terms of widths and heights, these units are actually completely idependent of the parent context; it does not matter how big the parent is or what scales it has, a physical size gives the same end result. These physical units are sensible to use when the parent is asking the child for its size.

The clear examples of relative units are native units and normalised parent coordinates. These depend on the size and scales of the parent to give a final size for the child. This sort of unit is **not** sensible to use when a parent is asking the child for its size because an infinite loop can be created; the size of the parent depends on the child, which depends on the parent, ...

There are two slightly odd cases. The first case involves `"lines"` and `"char"` units. These do depend on the parent in as much as the parent may have settings for the graphical parameters `lineheight` and `fontsize`. However, these are graphical parameter settings whereas the other things such as scales that define coordinate systems and the size of the parent form what I call the drawing context and structural description of the parent. What is important is to avoid asking a child for its size when the child depends on the drawing context (coordinate systems) or physical size of the parent. It is actually ok to have the child depend on the graphical parameters of the parent - this does not generate any sort of infinite loop.

The second odd case is `"null"` units. In this case, when the parent asks the child how big it should be, the child is effectively saying, "I don't know, give me whatever is left over after the others have had their share."

Pratical Implications

All of this stuff about absolute and relative units means that I would really only like `width.details` methods to return absolute units. For one thing, I don't want the method to return relative units because that could lead to weird results and it might lead to infinite loops. On the other hand, I *do* want the method to return absolute units because that can actually be quite handy (e.g., see the first example in this document).

Because I believe in giving the users plenty of rope to hang themselves, all I have done about this is provide a function which takes a unit and converts it to only absolute units; absolute units are untouched, but relative units get converted to `unit(1, "null")`. Consider the following example,

$\langle$*absolute*$\rangle\equiv$

```
grid.newpage()
gf <- grid.frame()
grid.pack(gf, grid.rect(width=unit(1, "inches"),
                        height=unit(.5, "npc"),
                        draw=F))
```

This means that the rectangle is packed into an area which is 1 inch wide, but as high as the parent decides it should be (based on what are is left after any other children have been accommodated), *and* that the rectangle actually only occupies half of the height that it is allocated.