Subject: Re: border around draw widget
Posted by JD Smith on Wed, 27 Sep 2006 22:23:36 GMT
View Forum Message <> Reply to Message

On Wed, 27 Sep 2006 23:35:34 +0200, Laurens wrote:

> Thanks very much for that explanation!
> Could you tell me how to make such a widget-object? It sounds like
> something I was already thinking about...

It sounds fancier than it is. It's basically an object, which:

1. Sets up a widget heirarchy in the normal way (usually in its Init method).
2. Saves its "state" information not in a structure in a UVALUE but in the class data itself (e.g. self.*).
3. Calls XManager (often, but not necessarily, in its Init method) to generate events on that widget heirarchy.
4. Uses the trick I outline to inject the events flowing forth from the widgets created to some class method (often named "Event").

The main advantages of this method:

1. You get state information "for free", quite nicely mapped to class data.
2. You automatically avoid common blocks for state info, with their associated collision risks if multiple identical widgets run at the same time.
3. You are never left with state information "in the air", if you use /NO_COPY to be efficient when retrieving your state structure from a UVALUE. This greatly aids debugging, since crashes to the code usually can be recovered from with a simple RETALL.
4. You quickly realize that the normal event flow embodied in "normal" widget prgramming is limiting, and can roll your own communication among objects that suits your needs. This is particularly useful if you have many different perhaps unrelated application components that need to communicate with eachother.

A schematic usage would be:

```
oDraw=obj_new('SelectableDrawPane',base)
```

which would place a new compound widget into base. It might implement some methods "Select" and "DeSelect", or you could have it trap the selection "clicks" and automagically select/deselect itself. Once you have the apparatus in place, you can then have fun implementing other methods for your object, drawing and erasing, etc.

I should note that none of this is necessary to use the "base on top of a base" trickery I outlined before, it just makes it easier and more powerful.

JD
