Subject: Re: Oddball Event Handling (Longer than it Ought to Be)
Posted by John-David T. Smith on Tue, 02 Jan 2001 17:22:44 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Hi David--
>
> Thanks for the cool description of you project.
>
> Now, prepare to be lightly toasted :-)
>
>> FUNCTION FindTLB, startID
>>
>> ; This function traces up the widget hierarchy to find the top-level base.
>>
>> FORWARD_FUNCTION FindTLB
>> parent = Widget_Info(startID, /Parent)
>> IF parent EQ 0 THEN RETURN, startID ELSE parent = FindTLB(parent)
>> RETURN, parent
>> END
>
> I have no problem with recursion. In this case however it's not
> really needed. For the book larnin' types, this is known as tail
> recursion I believe, which is often easily optimized. I admit
> recursion may help you conceptualize what's going on though. Wouldn't
> the following code do the same thing?
>
> parent = startid
> while widget_info(parent, /parent) NE 0 do $
> parent = widget_info(parent, /parent)
>

To defend the concept of recursion, you need a full tree walk. I found this lying among many other scraps, written several years ago. It goes the other direction, starting with a TLB (or any widget for that matter), and compiling a list of all widgets in the heirarchy beneath it. You can do it without recursion (as you can any algorithm), but it's ugly.

It's depth first (which, David, means that the recursive function call occurs before the part which looks "sideways" at a given tree level, so you descend all the way to the "leaf nodes" before unraveling the recursive stack). You can also easily make it breadth first search, so that siblings are discovered before children, grandchildren, etc. It's all in the ordering.

JD

```
;; decend heirarchy of tlb .. return entire tree's widget_ids in 'list'
pro treedesc, curin,list
```

```
  cur=curin                 ;ensure local copy of current widget
  if cur ne 0 then begin
    if n_elements(list) eq 0 then list=cur else list=[list,cur]
    cur=widget_info(cur,/CHILD) ;descend one level
    while cur ne 0 do begin ; find siblings... descend their subtrees
      treedesc,cur,list      ; recurs on sibling
      cur=widget_info(cur,/SIBLING)
    endwhile
  endif
end
```

File Attachments

1) treedesc.pro, downloaded 91 times