Subject: Re: IDLgrLegend Property Sheets, array properties
Posted by Chris[2] on Tue, 20 Apr 2004 19:14:58 GMT
View Forum Message <> Reply to Message

Hi Sean,

You basically have two options for handling non-scalar properties:

1. You can make it a type USERDEF, create an ::EditUserdefProperty method, and then handle the property on your own, using some sort of custom widget. This is good for really complicated properties.

2. However, as you suggested, you can also just split the property up into multiple properties.

If you do #2, you *must* explicitely handle these properties in the Get/SetProperty of your class, just like all other properties. You can't just do the replacement of the property value within the event handler, because this will only work for SetProperty. When the property sheet is displayed, it automatically calls ::GetProperty to retrieve the values. So you need to handle splitting the property up, and repackaging it, within your Get/SetProperty of your subclass.

This would work fine for a property that has a known # of values. Unfortunately, in your case, you have a problem because your legend could have an unlimited # of values, which makes it difficult to handle via keywords to Get/SetProperty.

We actually had the same problem in the iTools, with iContour. In this case, you can have multiple contour levels, each with its own set of properties. To solve this, we used a USERDEF property, that fires up a multi-column property sheet, with each column corresponding to a contour level. This *might* be a good solution. If you do this, you will need to create a lightweight "legend item" class, which contains the properties for a single item. You then hand an array of these legend item objects to the property sheet.

Or, a hacky solution would be to limit the # of legend items to say 10, and then just hardcode a bunch of keywords to your subclass Get/Setproperty.

Hope this helps.

-Chris
Research Systems, Inc.

"Sean Dettrick" <sdettrick@hotmail.com> wrote in message
news:c233c3ea.0404201018.5c9690d9@posting.google.com...

```
>  I hate to reply to my own message but it has occurred to me that I can
>  just register each element of the ITEM_NAME and each triplet of the
>  ITEM_COLOR as a seperate property with a seperate name (e.g.
>  ITEM_NAME0, ITEM_NAME1, etc).  In principle I guess the following
>  would work ( I say "in principle", because so far I can't get it to
>  work)
>
>  ; Register ARRAY properties element-by-element
>  for i=0,n_elements( item_names )-1 do begin
>     self -> IDLitComponent::RegisterProperty, $
>                  'ITEM_NAME'+strtrim(i,2),/STRING
>     self -> IDLitComponent::RegisterProperty, $
>                  'ITEM_COLOR'+strtrim(i,2), /COLOR
>
>     self -> IDLitComponent::SetPropertyByIdentifier, $
>                  'ITEM_NAME'+strtrim(i,2), item_names[i]
>     self -> IDLitComponent::SetPropertyByIdentifier, $
>                  'ITEM_COLOR'+strtrim(i,2), reform( item_colors[*,i]
>  )
>  endfor
>
>  Then the event handler could use SetProperty instead of
>  SetPropertyByIdentifier to make changes to the actual ITEM_NAME array
>  property:
>
>  new_value = WIDGET_INFO(event.id, $
>                       COMPONENT = event.component, $
>                       PROPERTY_VALUE = event.identifier)
>  if strmatch( event.identifier, 'ITEM_NAME*' ) eq 1 then begin
>     self -> IDLgrLegend::GetProperty, item_name=item_names
>     modify = fix( strmid( event.identifier, $
>                  strpos( event.identifer,'ITEM_NAME') $
>                       +strlen('ITEM_NAME')))
>     item_names[modify] = new_value
>     self -> IDLgrLegend::SetProperty, item_name=item_names
>  endif
>
>  By the way this is all inside a class which inherits from the
>  IDLgrLegend class.
```