Overview

The Hotspot class defines a regional hotspot, a transparent area that overlays one or more other objects. Regional hotspots can detect and respond to user selections with the mouse or ENTER key and run any associated SCL code when selected. In the bar chart shown here, the middle bar has been hotspotted with a regional hotspot. When users select this area, a series of programming tasks may occur.

Parent:

sashelp.fsp.widget.class

Class:

sashelp.fsp.Hotspot.class

Using the Hotspot Class

To be effective, a regional hotspot should be created on top of another object. The hotspot may be created on top of a portion of the object, or the hotspot may completely surround the object. With the exception of text entry objects, objects that are surrounded by a hotspot will not receive any events unless the hotspot is hidden with the _hide method. Therefore, a hotspot that is surrounded by another hotspot only works when the outer hotspot is hidden.
Defining Hotspots for Text Entry Objects

To define a hotspot for a text entry object, define the text entry fields as selectable by setting the Selection Style to **ENTER/single mouse click** in the Text Entry Attributes window. See the Text Entry class.

Changing Object Colors

Regional hotspots can contain color specifications to change the color of the object the hotspot overlays. For example, you can use a regional hotspot to define a selectable area on a large text entry object. When a user selects it, the area can change color and execute a program label to update a variable.

Note: If a regional hotspot fills a region that has a background color set, the background color disappears. If the region is subsequently emptied, the background color does not reappear; the region is transparent.

Manipulating the Hotspot in an SCL Program

Manipulate a hotspot in an SCL program by using its associated variable. The value of the hotspot variable is a list identifier for a list containing information about the most recently selected point on the object. You can test for the variable value in your SCL program, or you can determine the value with the `.getValue` method.

For example, assume a user selected one of the words in a text entry object. The list enables you to determine whether the user selected a word on the text entry object, and if so, which word it was and where it was on the text entry object. If you place a hotspot over a SAS/GRAPH output object, you can return the number of the specific graphics segment that was selected. (This value is stored in the SCL list item SPOTID.) Hotspots assigned to graphics output are called segment hotspots. See the SAS/GRAPH Output class for more information on creating and using segment hotspots.

Methods

Methods specific to the Hotspot class are described here. Inherited methods are described in the Object class and the Widget class.

Dictionary

_.getValue_

Returns the identifier of an SCL list containing descriptive information about the last point selected on the hotspot.
Syntax

CALL NOTIFY (hotspot-name, '_getValue', hot-list-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot-list-id</td>
<td>N</td>
<td>returns the identifier of an SCL list containing information about the location of the most recently selected area on the hotspot. The SCL list contains the following items:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL</td>
<td>N</td>
<td>the column of the X coordinate</td>
</tr>
<tr>
<td>OBJ NAME</td>
<td>C</td>
<td>the name of the selected object</td>
</tr>
<tr>
<td>ROW</td>
<td>N</td>
<td>the row of the Y coordinate</td>
</tr>
<tr>
<td>SPOTID</td>
<td>N</td>
<td>if the hotspot overlays a graph, the number of the graph segment within the graph, beginning with 1; otherwise, the value of SPOTID is -1</td>
</tr>
<tr>
<td>SPOTTYPE</td>
<td>N</td>
<td>a number identifying the type of graph segment selected.</td>
</tr>
<tr>
<td>TEXT</td>
<td>C</td>
<td>the text of the selected area</td>
</tr>
<tr>
<td>X</td>
<td>N</td>
<td>the X coordinate of the selection, in pixels</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>the Y coordinate of selection, in pixels</td>
</tr>
</tbody>
</table>

Example

Assume you have a FRAME entry that contains three buttons (labeled Default, Switch, and Hide) and a text entry object with two hotspots: a red hotspot covering the upper and lower right corners and a blue hotspot covering the upper and lower left corners. The text entry object was defined as selectable in the Text Entry Attributes window. In addition, there are two text entry fields containing the text selected in from each hotspot.

In the following example, when Switch is pressed, the red hotspot changes to blue and the blue hotspot changes to red. When Hide is pressed, the hotspots are no longer visible:

```
DEFAULT:
    /* Set up the default colors of the hotspots and display them. */
    call notify('hot1', '_show_hotspots_', 'red', 'reverse');
    call notify('hot2', '_show_hotspots_', 'blue', 'reverse');
    return;

SWITCH:
    /* Switch the currently displayed colors of the hotspots. */
```

```
call notify('hot1', 'show_hotspots', 'blue', 'reverse');
call notify('hot2', 'show_hotspots', 'red', 'reverse');
return;

HIDE:
   /* Hide the hotspots. */
   call notify('hot1', '_hide_hotspots_');
   call notify('hot2', '_hide_hotspots_');
   return;

HOT1:
   /* Put the clicked on word in HOTFLD1
and blank out HOTFLD2. */
   /* NOTE: The next line is not needed.
   It is just there to   */
   /* illustrate the _getValue method. */
   call notify('hot1', '_get_value_', hot1);
   hotfld1 = getnitemc(hot1, 'text');
   hotfld2 = _blank_
   return;

HOT2:
   /* Put the clicked on word in HOTFLD2
and blank out HOTFLD1. */
   /* NOTE: The next line is not needed.
   It is just there to   */
   /* illustrate the _getValue method. */
   call notify('hot2', '_get_value_', hot2);
   hotfld1 = _blank_
   hotfld2 = getnitemc(hot2, 'text');
   return;

_hideHotspots

Removes highlighting from all objects intersected by a hotspot

Syntax

CALL NOTIFY (hotspot-name, '_hideHotspots');

Details

When a hotspot's highlighting is removed, the highlighted area reverts to the object's color, and the hotspot outline is removed from the display.
---showHotspots

Highlights all objects intersected by a hotspot

Syntax

CALL NOTIFY (hotspot-name, '_showHotspots', <color, attr>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>C</td>
<td>specifies the highlight color</td>
</tr>
</tbody>
</table>
| attr     | C    | specifies the display attribute of the hotspot highlight:  
           |       | 'BLINKING' |
           |       | 'HIGHLIGHT'  |
           |       | 'HREV'  |
           |       | 'NONE' |
           |       | 'REVERSE' |
           |       | 'UNDERLINE' |

Details

If you omit the color and attr parameters, the color and attribute values specified in the Attributes window are used, and the region outline appears. Using this function, you can get a visible clue of which areas you may select. The color changes caused by this method are visible only over text entry and text label objects. Over all other objects, only the region outline becomes visible.

Example

_showHotspots highlights the hotspot and changes the color used from the default to blue reverse. HOT2 is displayed using the default color and attributes. For the color changes in the object to be seen, the hotspotted object must be of the Text Entry class. In any case, the region outline appears.

INIT:
    call notify('hot1','_show_hotspots_','blue','reverse');
    call notify('hot2','_show_hotspots_');
    return;