The SPOOR CORE

User Interface Classes for SPOOR Programming
Manual version $Revision: 2.6 $
$Date: 1995/12/13 01:25:50 $

Bob Glickstein

This manual documents The Spoor Core, a collection of Spoor user-interface classes.

Introduction

This manual describes The Spoor Core, a collection of user-interface classes for use in Spoor application programming.

1 Core Classes

Here is the hierarchy of the Spoor Core classes, with detailed programming information in the sections that follow. Superclasses are recursively shown in parentheses.

`spEvent (spoor)`
: An abstraction of a deferred event. Contains a time (relative or absolute) at which to trigger, and a function to call when the event triggers.

`spKeymap (spoor)`
: An abstraction for mapping between sequences of user keystrokes and user-interface actions to invoke.

`spFullKm (spKeymap (spoor))`
: A subclass of `spKeymap` implemented as a static-sized array containing entries for every possible single keystroke.

`spSparseKm (spKeymap (spoor))`
: A subclass of `spKeymap` implemented as a linked list of single keystrokes, for keymaps that are sparsely populated.

`spObservable (spoor)`
: A class of objects which can be observed, in the sense that interested observers can be notified of particular kinds of events.

`spButton (spObservable (spoor))`
: An object with a label; when pressed, it notifies its observers.

`spToggle (spButton (spoor))`
: A button with one bit of state which is toggled with each button press.

`spText (spObservable (spoor))`
: A dynamically-sized editable text buffer including position markers which move with the text as it changes.

`spView (spObservable (spoor))`
: The central class of the Spoor Core. A view subclass knows how to observe a particular kind of observable, and embodies the information necessary to display the object on the screen perform user interactions on the object. The *view/data separation* model permits, among other things, putting multiple views on a single data object.

`spButtonv (spView (spObservable (spoor)))`
: A view for the `spButton` class that permits the grouping of several buttons in several different layouts. A callback function is activated when one of the buttons is pressed.

`spMenu (spButtonv (spView (spObservable (spoor))))`
: A subclass of `spButtonv` that works like a pulldown or popup menu.

`spIm (spView (spObservable (spoor)))`
: A special subclass of `spView`, this is a class of *interaction managers*. There is one interaction manager object per application, and it has the ultimate responsibility for the appearance of the display and for coordinating input and output.

`spCharIm (spIm (spView (spObservable (spoor))))`
: A subclass of `spIm` for character-based applications.

`spCursesIm (spCharIm (spIm (spView (spObservable (spoor)))))`
: A subclass of `spCharIm` for curses-based implementations of character-based applications.

`spPopupView (spView (spObservable (spoor)))`
: A kind of view that can pop up on top of other views (and which can contain other views which wouldn't otherwise be able to pop up). Among other things, this is the view used to contain popup dialogs and pulldown menus.

`spSplitview (spView (spObservable (spoor)))`
: A view containing two subviews, either side-by-side or top-and-bottom.

`spTextview (spView (spObservable (spoor)))`
: A view for interacting with editable text.

`spCmdline (spTextview (spView (spObservable (spoor))))`
: A subclass of `spTextview` that is only one line high and has a callback associated with pressing the `RET` key.

`spWrapview (spView (spObservable (spoor)))`
: An instance of `spWrapview` contains one subview, and optionally adorns it in certain ways.

`spWindow (spoor)`
: This class represents a drawable region of the screen.

`spCharWin (spWindow (spoor))`
: This subclass of `spWindow` is for character-based screens.

`spCursesWin (spCharWin (spWindow (spoor)))`
: This subclass of `spCharWin` is for character-based screens controlled by curses.

Indentation is used below to reflect the inheritance hierarchy.

```
spEvent
spKeymap
    spFullKm
    spSparseKm
spObservable
    spButton
        spToggle
    spText
    spView
        spButtonv
            spMenu
        spIm
            spCharIm
                spCursesIm
        spPopupView
        spSplitview
        spTextview
            spCmdline
        spWrapview
spWindow
    spCharWin
        spCursesWin
```

1.1 spEvent

Superclass: `spoor`.

`void setup` (*long* `sec`, *long* `usec`, *int* `relative`, *int* (*`*func`)(), *void* *`data`) [Method on `spEvent`]

Initializes the data in *self*. *sec* and *usec* are seconds and milliseconds representing when this event should trigger. If *relative* is non-zero, then the seconds and milliseconds are considered to be a time in the future relative to now; otherwise they signify an absolute time. The `int`-returning callback function *func* is called when the event is triggered, and it takes two arguments: the event object itself, and the interaction manager object in whose event queue the event was queued. *func* should return zero if the event object should be simply forgotten by the interaction manager after the event is triggered, and non-zero if the interaction manager should actually destroy the event (using `spoor_DestroyInstance`) afterward. *data* is a pointer to arbitrary data; this pointer can be accessed using `spEvent_data`.

`int process` (*struct spIm* *`im`) [Method on `spEvent`]

This method is invoked by the main `interact` loop of an interaction manager, *im*, to process an event in its event queue. The value returned to the interaction manager is the value of the event's callback function and indicates whether the interaction manager is to destroy the event after processing it.

`int cancel` (*int* `destroy`) [Method on `spEvent`]

Cancel an event. The event remains in the interaction manager's event queue, but when `process` is invoked on the event, the callback function is not called. *destroy*

indicates whether the event should be destroyed by the interaction manager after it is processed.

`void * spEvent_data` (*struct spEvent *e*) [Accessor]
Yields the callback data associated with *e* in a call to the `setup` method.

`void * spEvent_inqueue` (*struct spEvent *e*) [Accessor]
Yields zero or non-zero depending on whether *e* is or isn't presently in some interaction manager's event queue. Note that this can be true even if the event has been canceled with `cancel`.

1.2 spKeymap

Superclass: `spoor`.

`spKeymap_none`
`spKeymap_function`
`spKeymap_keymap`
`spKeymap_removed`
`spKeymap_translation`
`spKeymap_undefined`

`spKeymap_XMIN`
`spKeymap_FKEYS`
`spKeymap_F(`*n*`)`
`spKeymap_sF(`*n*`)`
`spKeymap_cF(`*n*`)`
`spKeymap_F0`
`spKeymap_sF0`
`spKeymap_cF0`
`spKeymap_bTab`
`spKeymap_break`
`spKeymap_cTab`
`spKeymap_delete`
`spKeymap_down`
`spKeymap_end`
`spKeymap_home`
`spKeymap_insert`
`spKeymap_left`
`spKeymap_pageDown`
`spKeymap_pageUp`
`spKeymap_pause`
`spKeymap_printScreen`
`spKeymap_right`
`spKeymap_sTab`
`spKeymap_scrollLock`
`spKeymap_up`
`spKeymap_XKEYS`

`void addFunction` (*spChar *keys*, *void* (**fn*)(), *struct spoor *object*, *void *data*) [Method on `spKeymap`]

`void addKeymap` (*spChar *keys*, *struct spKeymap *keymap*) [Method on `spKeymap`]

`void addTranslation` (*spChar *keys*, *spChar *val*) [Method on `spKeymap`]

`void addUndefined` (*spChar *keys*) [Method on `spKeymap`]

`struct spKeymapEntry * lookup` (*spChar *keys*) [Method on `spKeymap`]

`void remove` (*spChar *keys*) [Method on `spKeymap`]

`struct spKeymap * copy` () [Method on `spKeymap`]

`char * spKeymap_charName` (*spChar ch*, *int pretty*) [Function]

`spChar spKeymap_nameChar` (*char *name*, *char **after*) [Function]

`int ci_strcmp` (*char *s1*, *char *s2*) [Function]

1.3 spFullKm

Superclass: `spKeymap` (see Section 1.2 [spKeymap], page 5).

1.4 spSparseKm

Superclass: `spKeymap` (see Section 1.2 [spKeymap], page 5).

1.5 spObservable

Superclass: `spoor`.

`spObservable_contentChanged`
`spObservable_destroyed`
`spObservable_OBSERVATIONS`

`void addObserver` [Method on `spObservable`]

`void notifyObservers` [Method on `spObservale`]

`void removeObserver` [Method on `spObservable`]

`void receiveNotification` [Method on `spObservable`]

`void setOwner` [Method on `spObservable`]

`int spObservable_numObservers` (*struct spObservable *obs*) [Accessor]

`struct spoor * spObservable_owner` (*struct spObservable *obs*) [Accessor]

1.6 spButton

Superclass: `spObservable` (see Section 1.5 [spObservable], page 6).

`void push` () [Method on `spButton`]

`void setLabel` (*char *label*) [Method on `spButton`]

`char * spButton_label` (*struct spButton *b*) [Accessor]

1.7 spToggle

Superclass: `spButton` (see Section 1.6 [spButton], page 6).

`int spToggle_state (struct spToggle *t)` [Accessor]

1.8 spText

Superclass: `spObservable` (see Section 1.5 [spObservable], page 6).

```
spText_linesChanged
spText_readOnlynessChanged
spText_OBSERVATIONS
```

`void clear` [Method on `spText`]

`void insert (int pos, int len, char *text, int afterMark)` [Method on `spText`]

`void delete (int pos, int len)` [Method on `spText`]

`void substring (int pos, int len, char *buf)` [Method on `spText`]

`int length` [Method on `spText`]

`int addMark (int pos, int after)` [Method on `spText`]

`void removeMark (int mark)` [Method on `spText`]

`void setMark (int mark, int pos)` [Method on `spText`]

`int markPos (int mark)` [Method on `spText`]

`void setReadOnly (int readonly)` [Method on `spText`]

`void writePartial (FILE *fp, int start, int len)` [Method on `spText`]

`void writeFile (char *filename)` [Method on `spText`]

`void readFile (char *filename)` [Method on `spText`]

`void fillDynstr (struct dynstr *d)` [Method on `spText`]

`int rxpSearch (regexp_t rxp, int pos, int *after)` [Method on `spText`]

`int spText_readOnly (struct spText *t)` [Accessor]

`int spText_newlines (struct spText *t)` [Accessor]

1.9 spView

Superclass: `spObservable` (see Section 1.5 [spObservable], page 6).

```
spView_fullUpdate
spView_parentUpdate
spView_UPDATEFLAGS
```

`void setObserved (struct spObservable *obs)` [Method on `spView`]

`void receiveFocus ()` [Method on `spView`]

`void loseFocus ()` [Method on `spView`]

`void wantFocus` (*struct spView* ***requestor***) [Method on `spView`]

`void wantUpdate` (*struct spView* ***requestor***, *unsigned long* `flags`) [Method on `spView`]

`void desiredSize` (*int* ***minh***, *int* ***minw***, *int* ***maxh***, *int* ***maxw***, *int* ***besth***, *int* ***bestw***) [Method on `spView`]

`void embed` (*struct spView* ***parent***) [Method on `spView`]

`void unEmbed` () [Method on `spView`]

`struct spIm * getIm` () [Method on `spView`]

`void install` (*struct spWindow* ***window***) [Method on `spView`]

`void unInstall` () [Method on `spView`]

`void overwrite` (*struct spWindow* ***window***) [Method on `spView`]

`void destroyObserved` () [Method on `spView`]

`void invokeInteraction` (*char* ***name***, *struct spoor* ***requestor***, *void* ***data***, *spChar* ***keys***) [Method on `spView`]

`spViewFn_t nextSubview` () [Method on `spView`]

`void wantNewSize` (*struct spView* ***requestor***, *int* `minh`, *int* `minw`, *int* `maxh`, *int* `maxw`, *int* `besth`, *int* `bestw`) [Method on `spView`]

`struct spObservable * spView_observed` (*struct spView* **v*) [Accessor]

`struct spView * spView_parent` (*struct spView* **v*) [Accessor]

`struct spWindow * spView_window` (*struct spView* **v*) [Accessor]

`struct spKeymap * spView_keymap` (*struct spView* **v*) [Accessor]

`view-invoke` [Interaction]

`void spView_addInteraction` (*struct glist* ***list***, *char* ***name***, *char* ***descr***, *void* (***func***)()) [Function]

`void spView_addInteractionByClass` (*spoorClass_t* ***class***, *char* ***name***, *char* ***descr***, *void* (***fn***)()) [Function]

`struct glist * spView_lookupInteractions` (*spoorClass_t* ***class***, *spoorClass_t* ****outclass***) [Function]

`(void (*) ()) spView_lookupInteraction` (*char* ***name***, *spoorClass_t* ***class***, *spoorClass_t* ****outclass***) [Function]

`struct spKeymap * spView_classKeymap` (*spoorClass_t* ***class***, *int* `full`, *int* `inherit`) [Function]

`struct spKeymap * spView_lookupKeymap` (*spoorClass_t* ***class***, *spoorClass_t* ****outclass***, *int* ***inherit***) [Function]

`void spView_bindKeymapKey` (*struct spKeymap *km*, *char *keys*, *void (*fn)()*, *struct spoor *obj*, *void *data*) [Function]

`void spView_bindInstanceKey` (*struct spView *self*, *char *keys*, *void (*fn)()*, *struct spoor *obj*, *void *data*) [Function]

`void spView_bindClassKey` (*spoorClass_t *class*, *char *keys*, *void (*fn)()*, *struct spoor *obj*, *void *data*) [Function]

`void spView_unbindKeymapKey` (*struct spKeymap *km*, *char *keys*) [Function]

`void spView_unbindInstanceKey` (*struct spView *view*, *char *keys*) [Function]

`void spView_unbindClassKey` (*spoorClass_t *class*, *char *keys*) [Function]

`void spView_BuildInteractionMap` () [Function]

`char * spView_InteractionName` (*void (*fn)()*) [Function]

`void spView_AssociateLabel` (*char *sequence*, *void (*fn)()*, *char *label*) [Function]

`char * spView_LookupLabel` (*spChar *sequence*, *void (*fn)()*) [Function]

1.10 spButtonv

Superclass: `spView` (see Section 1.9 [spView], page 7).

`spButtonv_grid`
`spButtonv_horizontal`
`spButtonv_multirow`
`spButtonv_vertical`
`spButtonv_STYLES`

`spButtonv_brackets`
`spButtonv_checkbox`
`spButtonv_inverse`
`spButtonv_STYLES`

`spButtonv_click`
`spButtonv_controlclick`
`spButtonv_shiftclick`
`spButtonv_CLICKTYPES`

`int spButtonv_anticipatedWidth` (*struct spButtonv *b*) [Accessor]

`struct spButton * spButtonv_button` (*struct spButtonv *b*, *int *num*) [Accessor]

`struct glist * spButtonv_buttons` (*struct spButtonv *b*) [Accessor]

`(void (*) ()) spButtonv_fn` (*struct spButtonv *b*) [Accessor]

`int spButtonv_length` (*struct spButtonv *b*) [Accessor]

`int spButtonv_selection` (*struct spButtonv *b*) [Accessor]

`enum spButtonv_style spButtonv_style` (*struct spButtonv *b*) [Accessor]

`enum spButtonv_toggleStyle spButtonv_toggleStyle` (*struct spButtonv *b*) [Accessor]

`struct spoor * spButtonv_obj` (*struct spButtonv *b*) [Accessor]

`int spButtonv_scrunch` (*struct spButtonv *b*) [Accessor]

`int spButtonv_clickMeansPush` (*struct spButtonv *b*) [Accessor]

`int spButtonv_highlightWithoutFocus` (*struct spButtonv *b*) [Accessor]

`void insert` (*struct spButton *b*, *int pos*) [Method on `spButtonv`]

`void remove` (*int num*) [Method on `spButtonv`]

`buttonpanel-left` [Interaction]

`buttonpanel-right` [Interaction]

`buttonpanel-up` [Interaction]

`buttonpanel-down` [Interaction]

`buttonpanel-search` [Interaction]

`buttonpanel-first` [Interaction]

`buttonpanel-last` [Interaction]

`buttonpanel-click` [Interaction]

`buttonpanel-shiftclick` [Interaction]

`buttonpanel-controlclick` [Interaction]

`buttonpanel-next-page` [Interaction]

`buttonpanel-previous-page` [Interaction]

`buttonpanel-glitch-up` [Interaction]

`buttonpanel-glitch-down` [Interaction]

`buttonpanel-click-by-name` [Interaction]

1.11 spMenu

Superclass: `spButtonv` (see Section 1.10 [spButtonv], page 9).

`spMenu_function`
`spMenu_menu`

`void addFunction` (*struct spButton *b*, *void* (**fn*)(), *int pos*, *struct spoor *obj*, *void *data*) [Method on `spMenu`]

`void addMenu` (*struct spButton *b*, *struct spMenu *m*, *int pos*) [Method on `spMenu`]

`struct spMenu * spMenu_superMenu` (*struct spMenu *m*) [Accessor]

`char * spMenu_label` (*struct spMenu *`m`, int `i`*) [Accessor]

`struct spMenu_entry * spMenu_Nth` (*struct spMenu *`m`, int `i`*) [Accessor]

`(void (*) ()) spMenu_cancelfn` (*struct spMenu *`m`*) [Accessor]

`menu-cancel` [Interaction]

`menu-left` [Interaction]

`menu-right` [Interaction]

`menu-down` [Interaction]

1.12 spIm

Superclass: `spView` (see Section 1.9 [spView], page 7).

`void setView` (*struct spView *`v`*) [Method on `spIm`]

`struct spKeymapEntry * updateKeystate` (*spChar *`keys`*, *spChar *`inbuf`*, *struct spView **`inview`*) [Method on `spIm`]

`void addTranslation` (*spChar *`keys`*, *spChar *`val`*) [Method on `spIm`]

`void addToFocusList` (*struct spView *`v`*, *int `afterp`*, *struct spView *`neighbor`*) [Method on `spIm`]

`void removeFromFocusList` (*struct spView *`v`*) [Method on `spIm`]

`void enqueueEvent` (*struct spEvent *`ev`*) [Method on `spIm`]

`void processEvent` () [Method on `spIm`]

`void message` (*char *`msg`*, *int `priority`*) [Method on `spIm`]

`void setFocusView` (*struct spView *`v`*) [Method on `setFocusView`]

`void resetFocuslist` () [Method on `spIm`]

`void popupView` (*struct spPopupView *`pv`*, *void* (**`fn`*)(), *int `desiredy`*, *int `desiredx`*) [Method on `spIm`]

`void dismissPopup` (*int `which`*) [Method on `spIm`]

`void watchInputFD` (*int `fd`*, *void* (**`fn`*)(), *void *`data`*) [Method on `spIm`]

`void unwatchInputFD` (*int `fd`*) [Method on `spIm`]

`void transientMessage` (*char *`msg`*, *int `priority`*, *int `duration`*) [Method on `spIm`]

`void dequeueViewUpdates` (*struct spView *`v`*) [Method on `spIm`]

`void forceUpdate` (*int `suppressSyncs`*) [Method on `spIm`]

`void forceDraw` () [Method on `spIm`]

`void refocus` () [Method on `spIm`]

`void processEvents` () [Method on `spIm`]

`interact-exit` [Interaction]

`interact-next` [Interaction]

`interact-previous` [Interaction]

`interact-redraw` [Interaction]

1.13 spCharIm

1.14 spCursesIm

1.15 spPopupView

1.16 spSplitview

1.17 spTextview

1.18 spCmdline

1.19 spWrapview

1.20 spWindow

1.21 spCharWin

1.22 spCursesWin

Index

R

S

T

U

V

W

Table of Contents