Frequently Asked Questions

In this section, we will answer the most popular SUIT related questions. The more involved answers are treated in great detail in the SUIT examples directory. Here you will find generously commented C programs that perform one function: the use of trappers, the use of menus, etc. The hope is that you will be able to copy the code directly into your own application with very little revision.

How Do I Create My Own Widgets?

There are several steps you need to take in order to create a widget of your own:

1.) Use **`SUIT_createObject()`** to create a new SUIT_object. You will provide a name for that widget and a new class that the widget will belong to. For example, you might create a new widget that implements a joystick-like control. Such a widget might be called “my joystick” and would belong to a class of widgets called “joystick”

2.) Use **`SUIT_addDisplayToObject()`** to register a display style for this new class of widget. All widget classes must have at least one display style. By convention, if a widget has but one display style, that style is called “standard”.

3.) Write a hit procedure for the widget. Such functions take the form:

```
void func (SUIT_object me, SUIT_event ev)
```

Hit procedures typically only set properties of a widget.

4.) Write a paint procedure for the widget. Such functions take the form:

```
void func (SUIT_object me)
```

Paint procedures typically use GP graphics calls to paint graphics based on the state of the widget’s properties. The widget’s property values are obtained through the SUIT_getProperty calls detailed onpage 60.

5.) If you want to be able to create your widget interactively, use **`SUIT_registerClass()`** to register the new widget class with SUIT. The creation proc this function takes as a parameter is the subject of the following note. Also see page 81.

NOTES:

1.) Usually, the **`SUIT_createObject()`** and **`SUIT_addDisplayToObject()`** calls are placed in a widget creation call of the form:

```
SUIT_object createNewWidget (char *name)
```

Notice that the function takes a string that can be used for this widget’s unique name and returns a **`SUIT_object`**. This will make your widget creation call as easy to use as SUIT’s own widget creation calls:

```
SUIT_object first, second;

first = createNewWidget (“first”);
second = createNewWidget (“second”);
```

2.) Paint procedures should NEVER set properties of a widget. When a widget property changes, the SUIT automatically flags the widget as needing a redisplay. This, in turn, will eventually cause the widget’s paint procedure to be called. Clearly, if a paint procedure is setting properties, this can cause the redisplay required flag to always be set which can send the application into an infinite loop. Remember, if a “SUIT_setXXX” calls doesn’t actually change a property value (i.e. it sets a property to the same value) then the redisplay flag is not set.

<u>**See Example File:**</u> **`NewWidgt.c`**

How do I use SUIT's menu widget?

See Example File: `menu.c`

How do I use SUIT's radio button widget?

See Example File: `radio1.c and radio2.c`

How do I use SUIT's scrollable list widget?

See Example File: `list.c`

How can I tell when a user has pressed a SHIFT-click?

See Example File: `events.c`

How do I create "HotKeys" for my buttons?

SUIT implements "hot keys" with something called a "trapper function" If you register a trapper function with SUIT using the **`SUIT_registerTrapper()`** call (see page 70), you can test for the occurrence of certain "special" keyboard and mouse events.

See Example File: `trapper.c`

What are "interests" and what are they good for?

Suppose you wanted to be sure that widget A always remained 10 pixels away from widget B. Wouldn't it be nice if you could be informed somehow every time the viewport property of widget B changed? That way, you could change A's viewport so that A followed B around. In SUIT parlance, we say that A "has an interest in" B. We use **`SUIT_registerInterest()`** to register an interest function that will get called every time any property of B changes. Interests are a simple way of enforcing constraints in SUIT.

See Example File: `interest.c`

How can I make one word in a label italic while the others are not?

GP_text supports a small, extensible notation for changing the attributes of text within a text string. This notation allows for underlines, italic, boldface, accents and even special characters like yen (¥) and pounds sterling (£) signs. You can also register your own accent marks using the function calls provided. For more information on this GP_text notation, see page 104.

See Example File: `events.c (look at the text box widget)`

How do I center text inside a button or label?

Set the JUSTIFICATION property of the label to center and SHRINK_TO_FIT to false.

How do I keep a property from being written to the SUI file?

Use **`SUIT_makePropertyTemporary()`**. See page 66.

How do I draw with pixels rather than GP's floating point numbers?

SUIT uses a floating point coordinate system entirely as a convenience. If you require access to the individual pixels of the screen, you can make calls to SRGP, the Simple Raster Graphics Package, which is SUIT's underlying graphics library. All of GP is written in SRGP. Documentation for SRGP is available as an addendum to the SUIT Reference Guide.

How do I make labels with more than one line of text?

Labels, by definition, can only display one line of text. To display several lines of text, use **`SUIT_createTextBox()`**. The text in this box is allowed to contain newlines as well as the special formatting characters detailed on page 104. For an example of **`SUIT_createTextBox()`**, see the file **`Events.c`**.

See Example File: **`events.c`**

How do I add "Info" help strings to the properties I create?

Use **`SUIT_registerHelp()`** to add help strings to SUIT's database of property information. You can retrieve help strings by using **`SUIT_getHelp()`**. Both functions are described on page 80.

How do I move / resize / remove the child widgets of a bulletin board?

Use "SUIT-o" to open the bulletin board (or any widget with children). A border will appear around the inside of parent widget, letting you know that the widget is open. Once the parent is open, you can SUIT-drag on the children to move and resize them. Use SUIT-k to close the bulletin board.

Why do the child widgets of my bulletin board show up in the right place, but the wrong size?

You're probably setting the SHRINK_TO_FIT property to TRUE and that's getting you into trouble. The widget gets sized with a **`SUIT_setViewport()`** call, but then the widget resizes to obey the SHRINK_TO_FIT property. This causes the widget's sides to collapse around the widget's center, making the widget the wrong size, according to the **`SUIT_setViewport()`** call. Solution: Set SHRINK_TO_FIT to be FALSE and throw away the VIEWPORT property in the property editor.

When can I make and destroy widgets in my code?

Widgets can be created at any time in your program after the call to **`SUIT_init()`**.

When is the ".sui" file read?

SUIT reads the ".sui" file in the call to **`SUIT_beginDisplay()`**. This function is called for you in the "convenience function" **`SUIT_beginStandardApplication()`**.

What are Locked Properties?

A property is said to be locked when the property cannot be changed in the SUIT property editor. Locking a property has the side effect of not allowing that property to be exported. You can lock and unlock properties under program control using the **`SUIT_lockProperty()`** and **`SUIT_unlockPropoerty()`** calls. Properties that are locked can still be changed under program control.

What are PERMANENT and TEMPORARY properties?

A property is permanent if the value of the property is to be written to the ".sui" file before the program exits. This way, the value can be restored when the program is next invoked. TEMPORARY properties are not written to the .sui file. See **`SUIT_makePropertyPermanent()`** and **`SUIT_makePropertyTemporary()`**.

Are SUIT object names case sensitive?

No.

Are SUIT property names case sensitive?

No.

Are SUIT type names, when they are represented as strings, case sensitive?

No.

What does "SUIT has detected an error..." mean? Is this a bug in SUIT?

Messages that look like:

```
* SUIT has detected an error at line 62 in the SUIT source file tree.c:
*
* SUIT_createObject was called with the name of an existing object
(`foo').
*
* In most cases, this type of error indicates that you have
* made a mistake when calling the SUIT library routines.
```

are almost always caused by errors in the application program, not in SUIT. For example, you might pass **`GP_rectangle()`** two points out of order (e.g. top right, bottom left, instead of bottom left, top right). Each kind of error comes with a detailed error message that should help you discover what caused the problem.

What do I do if I find a bug in SUIT?

By all means, report the bug.

It is important that you assume that we never heard of the bug so that we can have the fullest description of the problem.

Reports can sent via electronic mail to:

`suit@uvacs.cs.Virginia.EDU`

We appreciate any and all bug reports you might have. In order to consider your report, we'll need the following information:

- Machine Type (Sun, SGI, Mac, RS/6000, etc.)

- Color or Monochrome System

- A code snippet from the program that exhibits the problem.

- A detailed desctription of the way that the problem can be brought forth.

Can I Turn off the Property Editor?

Yes, see the section on shipping an application on page 161.

How do I force the display to act like monochrome if I'm on a color system?

Hand edit the ".sui" file and change the line that reads

```
#define THE_SCREEN_DEPTH  7
```

to

```
#define THE_SCREEN_DEPTH  1
```

How do I resize the application window?

(FOR WINDOWING ENVIROMENTS ONLY)
There are two ways to do this:

1.) You can resize the application window as you usually would (depends on the window manager, but typically involves dragging one corner of the window to a new location), then exit the applicaton. SUIT remembers the last size of the application window and uses it at startup.

2.) You can edit the SUI file directly. There are two lines at the beginning of the SUI file that read

```
#define THE_SCREEN_WIDTH <some number>
#define THE_SCREEN_HEIGHT <some other number>
```

By changing the numbers here, you can get a precise value for the size of the application window.

How do I create a bounded value with a textual readout?

See the example file provided with SUIT called BVLabel.c

See Example File: `bvlabel.c`

Type in Boxes keep coming up with the text from the last time I ran the program. How do I make sure that the type-in boxes are empty?

You need to make the CURRENT_VALUE property temporary, so it won't get written to the ".sui" file each time the program ends. To see how to do this,

See Example File: `tempprop.c`

How do I make a panel of widgets appear and disappear under program control?

Be careful here. Making widgets come and go can lead down the dark and dangerous path to the dreaded "moded interface." A moded interface, loosely speaking, is one where the user is prevented from taking certain actions because certain other actions are taking place. For example, imagine a drawing/text-editing application with two panels of widgets -- one for drawing graphics and one for editing text. It would be a poor interface indeed if the user was prevented from bringing up both panels of widgets at once. If bringing up one panel caused the other one to vanish, the interface would be "moded."

This kind of "one at a time" interaction is attractive from a programming point of view in that it seems to make the program easier to code (in fact, it rarely does), but it is very difficult and confusing to *use* a program written this way. If you're programming this way, you might want to re-read the tutorial appendix article, "An Introduction to External Control."

SUIT provides a number of dialog box calls that will help you with some common "safe" moded interactions (e.g. getting an OK confirmation from the user for a dangerous action). If you absolutely must make panels come and go under program control, you can refer to the file called

See Example File: `moded.c`

What does "SRGP: Color Table Too Full to share" mean?

It means that there is some other application running that uses a lot of colors (a complex background with a lot of colors is a common culprit). Unless you tell SUIT otherwise, SUIT will attempt to allocate 7 bitplanes for the SUIT color table (which is maintained by the SRGP graphics package). This allocates $2^7 = 128$ colors for you, but in the presence of another color-hungry application, there may not be that many free colors available.

The solution is to hand edit the ".sui" file, changing the lines that read

```
#define THE_SCREEN_DEPTH  7
```

to

```
#define THE_SCREEN_DEPTH  <some number lower than 7>
```

What if the widget I am editing covers the whole Property Editor?

Press SUIT-B to send the widget to the back (behind the property editor).

How Do I set A Property to a Default Value?
How Do I keep the .sui File from Resetting a Property?

The ".sui" file is read at the beginning of the call to **`SUIT_beginStandardApplication()`**. If you need to reset some properties after the ".sui" file is read, you can use the expanded version of the SUIT main loop like so:

```
SUIT_beginDisplay();
/* Set widget defaults here */
SUIT_setBoolean(my_obj, HAS_BORDER, FALSE);
while (TRUE) {
  SUIT_checkAndProcessInput();
}
```

Of course, you can always call SUIT_makePropertyTemporary() to keep the property from being written to the ".sui" file at all.

What are the SUIT-command keys?

Below is a complete list of all the SUIT command keys:

Operation	**What it Does**	**Hot Key**[1]
SUIT menu	*invokes the SUIT menu, which contains most of the following functions*	SUIT-M
cycle	*change a widget's display style*	SUIT-C
align	*lines up selected widgets by tops, bottoms, etc.*	SUIT-A
send to back	*selected widget goes behind all others*	SUIT-B
bring to front	*selected widget goes in front of all others*	SUIT-F
select widget	*marks a widget as selected; deselects a widget if already selected; selects all widgets if cursor is over no widget*	SUIT-S
redraw	*repaints all widgets*	SUIT-R
edit properties	*examine and alter a widget's properties*	SUIT-E
get info	*prints information about a widget in a dialog box*	SUIT-I
open widget	*opens up a parent widget so that the children may be accessed*	SUIT-O
close widget	*closes a parent widget that was opened with SUIT-o*	SUIT-K
create new widget	*creates a new SUIT object on the fly*	SUIT-N
destroy	*destroys a SUIT widget*	SUIT-X
version	*prints the version of SUIT you are using*	SUIT-V
duplicate	*creates a copy of a SUIT widget*	SUIT-D
quit application	*save the ".sui" file and leave the application.*	SUIT-Q
write ".sui" file	*write the sui file without exiting*	SUIT-W

1. "SUIT" is shorthand for holding down the SHIFT and CONTROL keys simultaneously.