The SUIT Main Loop

Every SUIT program will need to have the following calls in order to initialize SUIT and to get the SUIT main loop running. The lines are numbered and explained in the sections below:

A typical **main()** would look like:

```
    void main (int argc, char *argv[]) {
(1)     SUIT_init (argv[0]);
(2)     SUIT_beginDisplay();
            while (TRUE) {
(3)         SUIT_checkAndProcessInput(INDEFINITE)
        }

    }
```

Note that instead of the **while()** loop and a call to **SUIT_checkAndProcessInput()**, you can use a single call to **SUIT_beginStandardApplication(),** which is just a sugar coating for the above while loop. Below is explained the things that go on at each stage of the loop.

1.) SUIT_init (argv[0])

1.) Initializes all global variables that SUIT needs, including the name of the application.

2.) Registers all of SUIT's supported types (see page 35 for a discussion of types).

3.) Initializes all of the SUIT global properties (see page 110 for a list of global properties).

4.) Peeks at the .sui file to establish the correct screen size and window size.

5.) Begins the underlying raster graphics package, SRGP.

6.) Initializes the cursors, color table and fonts that SUIT uses.

7.) Initializes all the built-in widget classes (see page 108 for a list of SUIT widget classes).

2.) SUIT_beginDisplay()

1.) Reads the .sui file. Create any interactively created objects (those not created from the source code).

2.) Clears the screen.

3.) Draws all widgets for the first time.

3.) SUIT_checkAndProcessInput (timeToWait)

1.) REDISPLAY WIDGETS AS NEEDED: This step performs a redisplay of all widgets that have been flagged as requiring redisplay. Things that cause this flag to be set for a widget are:

- Changing the value of a property for that widget (See "When Do I Paint?", below). SUIT is careful not to flag the widget as needing a repaint if the result of setting a property does not in fact change the value.
  For example, if the CURRENT_VALUE of a bounded value is 3.141, calling
  `SUIT_setDouble (obj, CURRENT_VALUE, 1.0)`
  would flag the widget as needing a repaint, but
  `SUIT_setDouble (obj, CURRENT_VALUE, 3.141)`
  would not.
- Bringing the widget to the front of the stack of widgets
- Moving or resizing a widget
- Animating a widget

2.) GET EVENT: This step determines the type of the event that has occurred. Events are queued in a first-in, first- out manner. The timeToWait parameter determines the length of time in "ticks" (1/60 second) this function will wait for an event to enter the queue. If, after that period of time, no event has entered the queue, **`SUIT_checkAndProcessInput()`** exits. The one exception to this is if the **`timeToWait`** parameter is set to INDEFINITE, in which case, the function sleeps, waiting for an event. When an event finally does occur, the function exits.

3.) IS IT A SUIT EVENT? If the event happened while the SHIFT and CONTROL keys were pressed, the event is interpreted as a SUIT command and is processed as such (move, resize, open, etc.).

4.) HANDLE TRAPPER FUNCTION: If no trapper function has been registered, proceed to step 5, otherwise call the trapper function. Trapper functions designed to catch all input events before they are handed off to whatever widget was hit. Typical applications of trapper functions include adding hot keys to a menu and allowing the return key to activate a dialog box "OK" button, even if the cursor is not over the OK button. Trappers are discussed in detail on page 70. If the trapper returns NULL this means that the trapper has consumed and handled the event; we abort the loop and begin again. Otherwise, the trapper will return with a SUIT_object, though not necessarily the SUIT_object that was hit. This SUIT_object returned is handed over to step 5.

5.) HIT WIDGET: The hit procedure for the widget is called.

When Do I Paint?

1.) Painting a widget happens only when the widget has been flagged as requiring a repaint. Calling **`SUIT_redisplayRequired(obj)`** does not actually repaint the object, it merely flags the object as needing a repaint. The repainting happens at the beginning of each iteration of the SUIT main loop. If there is more than one widget that requires repainting (as there often is), the widgets are drawn from back to front in the Z-ordering of widgets.

2.) If you use any of the SUIT_set functions to change the value of a widget's property, SUIT will flag that widget as needing a repaint. Because of this "courtesy" side-effect, **it is vital that you not call any of the SUIT_setProperty functions from inside a widget's painting procedure.** Doing so will cause a widget to repaint multiple (perhaps endless) times because after each invocation of the paint procedure the widget will be flagged as needing redisplay, which will cause yet another call of the paint procedure.