NIRSPEC

UCLA Astrophysics Program **U.C. Berkeley** **W.M.Keck Observatory**

Fred Lacayanga

September 8, 1998

NIRSPEC Software Programming Note 30.00
Linking Dataviews Buttons/Menu Items to the Client

Introduction

This document describes the procedure for adding buttons, dynamic objects and menu items to the NIRSPEC client program and assumes knowledge of how use the DVdraw package to create those objects. All of NIRSPEC's Dataview files can be found in the `/kroot/kss/nirspec/ui/xnirspec` directory under the `/drawings` , `/layouts` , and `/views` subdirectories.

The naming convention for all the files in the `/views` subdirectory is:

(Type)_(Screen)_(Button)_(Top level menu item)_(Level 2 menu item)_(Level n menu item)

Type:

View	Display window or pop up window requiring input.
Menu	Pop up menus
Btn	Button

Screen:

Inst	Instrument Display
Scam	SCAM
Spec	Spectrometer

Button:

Eng	Engineering
Filter	Filter
Help	Help
Irot	Image rotator
Lamp	Calibration lamp
Resmode	Resolution mode
Setup	Setup
Slit	Slit Wheel
Slitview	SCAM
File	File
Sampmode	Sampling mode
Setup	Observation setup

Menu Item:

Button dependent

For example, the file: `menu_inst_eng_tsplink_write.v` refers to the DataViews view file of the write menu item, of the tsplink top level pop up menu item of the engineering button in the instrument display window.

Procedure for adding buttons

- Create a button using DVDraw naming the object using the naming convention above

  For example, the `Setup` button in the instrument display window has the object name

  ```
  btn_inst_setup
  ```

- Check for button press in the `handle_button_press()` function in `dataviews.c`

  This piece of code checks to see if the right mouse button was pressed. If so, then cancel the input and delete all pop up menus:

  ```
  if ( VOloButton( location ) == 3 ) {                /* Cancel */
          for ( i = 0; i < NUM_INPUT_OBJECTS; i++ ) {
              popup_delete( i );
          }
  }
  ```

  If it is not a cancel action, then determine if the action was in a valid location, i.e. button or menu item. This is done by string comparing the object's name to valid object names.

  ```
  else {
   object = TloGetSelectedObject (location);
      if ( object ) {
       if ( S_STRCMP( obj_name, "btn_inst_setup" ) == 0 ) {
                   popup_draw(MENU_INST_SETUP, drawport, TRUE);
       }
   }
  }
  ```

  Therefore, to add code to act upon an action on the new button, add the statement

  ```
       if ( S_STRCMP( obj_name, "new_button_name" ) == 0 ) {

                   /* insert code for button action */

       }
  ```

Procedure for adding dynamic objects

- Create a dynamic object and link it to a DataViews variable.

  For example, the transputer status message at the bottom of the instrument display window is a text object that is dynamically linked to the `status_message` variable.

- Define a client program variable in `xnirspec.h` that will link to the DataViews variable

  ```
  char VdpBuf_status_message[80]
  ```

- Link the client program variable to the DataViews variable in `dataviews.h`

  ```
  /*
   *  Symbol table to hold variable names and data info
   */
  static DATA_INFO data_table[] = {
  ```

```
"status_message",            (ADDRESS)&VdpBuf_status_message,

}
```

- If the dynamic object acts on a NIRSPEC keyword, then create interest in the keyword in `create_interest.c`

  ```
  EXPRESS_INTEREST(  "tspstat",   tspstat_callback );
  ```

  and add a callback function in `callbacks.c`

  ```
  void tspstat_callback( keyword, user_data, call_data, context )
  char *keyword;                     /* keyword                          */
  void *user_data;                   /* unused                           */
  KTL_POLYMORPH *call_data;          /* contains new value               */
  KTL_CONTEXT *context;              /* command context (unused)         */
  {

      if ( (strcmp( call_data->s, "IROT motor move success.") != 0) ||
           (strcmp( call_data->s, "IROT motor is busy.") != 0) ) {
                strcpy( VdpBuf_status_message, call_data->s );
                DV_updateScreen( SCREEN_INST );
      }
  }
  ```

Procedure for adding pop up menus

- Create motif menus with the appropriate object name

  ```
  menu_inst_eng
  ```

- Define C program variable in `xnirspec.h` that will contain the values passed back from the motif menu

  ```
  char VdpBuf_btn_inst_eng
  ```

- Link the client program variable to the DataViews variable in `dataviews.h`

  ```
  /*
   *  Symbol table to hold variable names and data info
   */
  static DATA_INFO data_table[] = {
  "btn_inst_eng",             (ADDRESS)&VdpBuf_btn_inst_eng,
  }
  ```

- Define new index for input object in `dataviews.h`

  ```
  #define menu_inst_eng    1
  ```

- In `dataviews.h` , define a function prototype for the menu callback that will be executed on input from the menu.

  ```
  static int menu_inst_eng_callback();
  ```

- Add info about menu item in `input_objects[ ]` array in the position corresponding to the index that was defined

View name	menu_inst_eng.v
Object name	menu_inst_eng
C program variable	VdpBuf_btn_inst_eng-
Callback function	menu_inst_eng_callback

```
static POPUP_INFO input_objects[] = {
0, NULL, "menu_inst_eng.v",         "menu_inst_eng",
    (ADDRESS)&VdpBuf_btn_inst_eng,      menu_inst_eng_callback,FALSE,
}
```

- Check for button press in the `handle_button_press()` function in `dataviews.c`

```
else if ( S_STRCMP( obj_name, "btn_inst_eng" ) == 0 ) {
    popup_draw( MENU_INST_ENG, drawport, TRUE );
}
```

- Add callback function in `dataviews.c`

```
INT menu_inst_eng_callback(
    OBJECT              index,
    EVENT_REQUEST       request,
    INT                 action,
    OBJECT              loc_event,
    ADDRESS             buffer )
{
    /*
     *  Delete all "MENU_INST_ENG_*" popups first
     */
    popup_delete( MENU_INST_ENG_MOTOR );
    popup_delete( MENU_INST_ENG_TSPLINK );
    popup_delete( MENU_INST_ENG_CLOCK );
    popup_delete( MENU_INST_ENG_CLOCK_SPEC );
    popup_delete( MENU_INST_ENG_CLOCK_SCAM );
    popup_delete( MENU_INST_ENG_OFFSETS );

    TdpDrawNext( input_objects[index].drawport );

    if (VdpBuf_btn_inst_eng == 1) {
        EngPasswd = FALSE;
        popup_delete( index );
}
    else if (VdpBuf_btn_inst_eng == 2 ) {
        popup_draw( MENU_INST_ENG_MOTOR, input_objects[index].drawport,
FALSE);
    }
    else if (VdpBuf_btn_inst_eng == 3 ) {
        popup_draw( MENU_INST_ENG_TSPLINK, input_objects[index].drawport,
FALSE);
    }
    else if (VdpBuf_btn_inst_eng == 4 ) {
        popup_draw( MENU_INST_ENG_CLOCK, input_objects[index].drawport,
FALSE);
    }
    else if (VdpBuf_btn_inst_eng == 5 ) {
        popup_draw( MENU_INST_ENG_OFFSETS, input_objects[index].drawport,
FALSE)
;
    }
    else if (VdpBuf_btn_inst_eng == 6) {
        popup_delete( index );

        if ( ktl_read( khand, KTL_WAIT, "lsclient", 0, &data, 0 ) < 0 )
            get_message( "Failed to get client info.", SCREEN_INST );
        else
            get_mmessage( "Nirspec Clients (user@host:port)", data.s,
SCREEN_INS
T );
}
    else
        popup_delete( index );
}
```