Resources and Resource Compilation

A Windows program stores its fonts, icons, cursors, bitmaps, and dialog boxes as resources. All the resources your application uses are described in a resource definition script. I normally name my resource file resources.rc. I add a line to the makefile to compile the resources with the windres resource compiler. The output is an object file which is merged into the application during the link phase.

I have added parts of the resources for S4L4 quadBounce here as an example. The resource.h file is #included into resource.rc. resource.h keeps track of name, number pairs. This is a translation file between the numbers used by the resource compiler and the text preferred by humans. The prefix to any resource define gives a clue as to how and where it is used. IDI for icons, IDR for accelerators, IDM for menus, IDD for dialog, and IDC for controls.

From resource.h

```
// glue indices
#define IDI_APPICON          101
#define IDI_SPAREICON        102
#define IDR_MAINMENU         103
#define IDR_ACCELERATOR      104

// menu choices
#define IDM_FILE_EXIT       4001
#define IDM_HELP_ABOUT      4002
#define IDM_FILE_RUN        4003
#define IDM_FILE_PAUSE      4004
#define IDM_PARAMETERS       4007
#define IDM_HELP_HELP       4008

// dialog choices
#define IDD_ABOUT_DIALOG     118
#define IDD_HELP_DIALOG      120
#define IDD_FIXED_DIALOG     121
#define IDD_RANDOM_DIALOG    122
#define IDD_MIXED_DIALOG     123
#define IDD_PARAM_DIALOG     124
#define IDD_CHOICE_DIALOG    125

// dialog control choices
#define IDC_RADUP            5001
#define IDC_RADIUS           5002
#define IDC_RADDOWN          5003
```

resource.rc uses the name, number pairs from resource.h then adds the windows.h header file. We start by hooking our Bacona Design icon into the application. That is followed by our two menus: File and Parameters. Menu item Run shows how you gray out the button. In this case Run is not implemented so it cannot be triggered. One day <sigh>. Pause is also grayed. I put a check mark next to the menu item Settings in the Parameters menu to show how that is done. Next, I show a few of the accelerator keys you can use as short cuts to the about, help, and parameter dialog boxes, and to exit the app.

From resource.rc

```
#include <windows.h>
#include "resource.h"


// Win32 application icon.
IDI_APPICON ICON    "BDc.ico"

// Our main menu.
IDR_MAINMENU MENU
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "Run",              IDM_FILE_RUN,   GRAYED
    MENUITEM "Pause",            IDM_FILE_PAUSE, GRAYED
    MENUITEM SEPARATOR
    MENUITEM "E&xit",            IDM_FILE_EXIT
  END

  POPUP "&Parameters"
  BEGIN
    MENUITEM "&Settings",         IDM_PARAMETERS, CHECKED
  END


// Our accelerators.
IDR_ACCELERATOR ACCELERATORS
BEGIN
  "A",           IDM_HELP_ABOUT,    VIRTKEY, ALT, NOINVERT
  "H",           IDM_HELP_HELP,     VIRTKEY, ALT
  "S",           IDM_PARAMETERS,    VIRTKEY, ALT
  "X",           IDM_FILE_EXIT,     VIRTKEY, ALT
END
```

I have included the introductory “splash” dialog box too. These are all the details the dialog procedure needs to do its job. Set up the window styles and caption, then set up the radio buttons, the text, and the OK button.

```
IDD_CHOICE_DIALOG DIALOGEX 0, 0, 171, 95
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "Population choice"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    CONTROL         "Fixed size",IDC_FIXED,"Button", BS_AUTORADIOBUTTON,
20,18,47,10
    CONTROL         "Mixed sizes", IDC_MIXED,"Button", BS_AUTORADIOBUTTON,
20,37,52,10
    CONTROL         "Random sizes",IDC_RANDOM, "Button", BS_AUTORADIOBUTTON,
20,55,59,10
    DEFPUSHBUTTON   "OK",IDOK,62,74,50,14
    LTEXT           "Please choose a ",IDC_STATIC,96,25,54,8
    LTEXT           "population type.",IDC_STATIC,96,36,54,8
END
```

Here is the description of one of the dialog boxes called from the choice dialog box. This is the screen you’ll see often as you modify the parameter settings during a run. Once again, we set up the window styles and caption. Then we add each of the various control buttons with their associated text. This dialog does not understand typed input to modify parameter settings. Rather, it uses up and down push buttons to control the settings.

```
IDD_PARAM_DIALOG DIALOGEX 0, 0, 280, 139
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "Parameters"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,147,109,50,14
    PUSHBUTTON      "Larger",IDC_CONDUCTUP,17,33,50,14
    LTEXT           "Static",IDC_CONDUCTION,33,56,30,8
    PUSHBUTTON      "Smaller",IDC_CONDUCTDOWN,17,70,50,14
    PUSHBUTTON      "Larger",IDC_AMBIENTUP,81,33,50,14
    LTEXT           "Static",IDC_AMBIENT,99,56,30,8
    PUSHBUTTON      "Smaller",IDC_AMBIENTDOWN,81,70,50,14
    PUSHBUTTON      "Larger",IDC_HEATUP,145,33,50,14
    LTEXT           "Static",IDC_HEAT,165,56,19,8
    PUSHBUTTON      "Smaller",IDC_HEATDOWN,145,70,50,14
    LTEXT           "Conduction",-1,24,22,37,8
    LTEXT           "Ambient",-1,91,22,27,8
    LTEXT           "Added Heat",-1,151,22,39,8
    PUSHBUTTON      "Larger",IDC_TIMEUP,209,33,50,14
    LTEXT           "Static",IDC_TIME,221,57,19,8
    PUSHBUTTON      "Smaller",IDC_TIMEDOWN,209,70,50,14
    LTEXT           "Time",-1,225,22,16,8
END
```

Here is the mixed population dialog box. Instead of labeling the push buttons with words I used < and > instead. This let me shrink them considerably and you only need one line of text for each parameter, flanked by the two buttons. Thus the current < parameter > value box format. Since each present value box is an EDITTEXT you can type directly into each of them and change the setting that way. But I found the < button 'parameter name' > button, value box, display format compact and descriptive.

```
IDD_MIXED_DIALOG DIALOGEX 0, 0, 198, 128
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "Mixed size population"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    EDITTEXT        IDC_SBALLS,17,23,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_MBALLS,18,49,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_LBALLS,18,75,40,14,ES_AUTOHSCROLL
    PUSHBUTTON      "<",IDC_SRADDN,158,24,9,9
    PUSHBUTTON      ">",IDC_SRADUP,174,24,9,9
    PUSHBUTTON      "<",IDC_MRADDN,158,48,9,9
    PUSHBUTTON      ">",IDC_MRADUP,174,48,9,9
    PUSHBUTTON      "<",IDC_LRADDN,158,75,9,9
    PUSHBUTTON      ">",IDC_LRADUP,174,75,9,9
    DEFPUSHBUTTON   "OK",IDOK,79,107,50,14

    LTEXT           "# Balls",IDC_STATIC,24,7,22,8,0,WS_EX_RIGHT
    LTEXT           "Small",IDC_STATIC,73,23,17,8
    LTEXT           "Medium",IDC_STATIC,74,49,25,8
    LTEXT           "Large",IDC_STATIC,75,75,19,8
    EDITTEXT        IDC_SRAD,106,21,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_MRAD,106,46,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_LRAD,106,73,40,14,ES_AUTOHSCROLL
    LTEXT           "Radius",IDC_STATIC,112,7,22,8
END
```

The makefile requires a few new lines. RC = windres to choose the windows resource compiler. The link rule needs resource.o to include it in the executable. The resource compilation rule requires resource.rc, the application manifest, our icon, and the resource.h file. The application manifest is an XML file which identifies the parts to bind at run time. Mine has a list of supported Windows operating systems, a version of Win32, and the version of Windows common controls. If you have problems with resource compilation, simplify the rule. Use only resource.rc and resource.h to see if they compile. Formatting in the makefile and in resource.rc is important. Indentations are one tab NOT three spaces. That causes a few errors. It is best to show any hidden characters while you are editing so you can debug your files. You may find tabs which have been implemented as a series of spaces. This is not correct.

Makefile changes

```
CC = g++
RC = windres
APP = convectTM

bounce: bounce.o ball.o resource.o
   ${CC} -mwindows -o ${APP} bounce.o ball.o resource.o ${LSFLAGS}

resource.o: resource.rc Application.manifest BDc.ico resource.h
   ${RC} -i resource.rc -o resource.o
```

Once you have all the parts in place, and working, you will create an application with menus and dialog boxes which is easy to expand and maintain. GUIs create a lot of drudge work, but using them does impress people. The very same app could run via command line but now you have a graphics window, menus, dialog boxes, and controls. This lets you modify your program's behavior on the fly by tweaking parameters using menus, accelerator keys, and dialog boxes.