```
/*****************************************************************************

    ArgusHelp.cp

*****************************************************************************/

/*
    This library provides a simple function call to display a dialog
    which displays some help or general information about the
    application.   Reads text from TEXT resource ID identified by
    HELP_TEXT.

    Functions Include:

      MyHelpDialog              Standard Argus Help Dialog

      // Semi-private
      MyHelpEventFilter
      SetUpHelpTEWindow
      MyHelpDoContent
      FnTE_AdjustText
      FnTE_SetToCursorPos
      FnTE_ScrollProc
*/


/********** Includes */
#include <Dialogs.h>
#include "Fn_Prototypes.h"

#include "MainWindow.h"      // may need to modify


/********** Defines */
#define HELP_DLOG        598
#define HELP_TEXT        598
#define NIL_PTR          0L
#define ALLOCATE_MEM     0
#define IN_FRONT         (WindowPtr)-1L
#define OK_BUTTON        1
#define PR_BUTTON        2
#define SV_BUTTON        3
#define RETURN_KEY       13
#define ENTER_KEY        3
#define VISUAL_DELAY     8   // standard is 8 ticks


TEHandle                 te = 0L;
ControlHandle            vScroll = 0L;
extern TEHandle          *tempTEHptr;
extern ControlHandle *tempVScrollptr;
Str255                   okButtonStr;
Str255                   printButtonStr;
Str255                   saveButtonStr;


/********** Prototypes */
Boolean          MyHelpDialog( void );
pascal Boolean MyHelpEventFilter(   DialogPtr       theDialog,
                                    EventRecord     *theEvent,
                                    short           *itemHit );
Boolean          SetUpHelpTEWindow(   WindowPtr       w,
```

```
                                    Boolean         hasGrowIcon,
                                    Boolean         wordWrap,
                                    short           font,
                                    short           textSize,
                                    short           textInsetPixels,
                                    // results //
                                    TEHandle        *te,
                                    ControlHandle *vScroll );
void            MyHelpDoContent(    WindowPtr       w,
                                    EventRecord     *e,
                                    TEHandle        *te,
                                    ControlHandle *vScroll,
                                    short           *itemHit );
extern void     FnTE_AdjustText(    TEHandle        *te,
                                    ControlHandle *vScroll );
extern void     FnTE_SetToCursorPos(TEHandle        *te,
                                    ControlHandle *vScroll );
extern pascal void FnTE_ScrollProc(ControlHandle theControl,
                                    short           theCode );


/********** MyHelpDialog */

Boolean MyHelpDialog( void )

    WindowPtr       docWindow;
    DialogPtr       dialog;
    Boolean         done;
    Boolean         result;
    short           itemHit;
    short           itemType;
    Handle          itemHandle;
    Rect            itemRect;
    EventRecord     theEvent;
    long            finalTicks;
    THPrint         pPrintH;
    short           vRef;
    short           refNum;

    result = FALSE;
    pPrintH = (TPrint **)NewHandle( sizeof( TPrint ) );
    PrintDefault( pPrintH );

    docWindow = FrontWindow();
    if( docWindow != NIL_PTR )
        MyDoDeactivateWindow( docWindow );

    dialog = GetNewDialog( HELP_DLOG, ALLOCATE_MEM, IN_FRONT );

    if( dialog == NIL_PTR )
        return( result );

    /* AdjustMenus_(); */
    SetUpHelpTEWindow( (WindowPtr)dialog, FALSE, TRUE, monaco, 9,
        4, &te, &vScroll );
    FnTE_GetTEXT( HELP_TEXT, &te, &vScroll );

    ShowWindow( dialog );
    FnTE_UpdateWindow( (WindowPtr)dialog, &te, FALSE );
    FnTE_FrameRect( (WindowPtr)dialog, &te, 4 );
    FnMisc_FrameButton( dialog, OK_BUTTON );
```

```
    GetDItem( dialog, OK_BUTTON, &itemType, &itemHandle, &itemRect );
    GetCTitle( (ControlHandle)itemHandle, okButtonStr );
    GetDItem( dialog, PR_BUTTON, &itemType, &itemHandle, &itemRect );
    GetCTitle( (ControlHandle)itemHandle, printButtonStr );
    GetDItem( dialog, SV_BUTTON, &itemType, &itemHandle, &itemRect );
    GetCTitle( (ControlHandle)itemHandle, saveButtonStr );

    done = FALSE;
    while( done == FALSE )

        ModalDialog( &MyHelpEventFilter, &itemHit );

        switch( itemHit )

            case OK_BUTTON:
                result = TRUE;
                done = TRUE;
                break;
            case PR_BUTTON:
                result = TRUE;
                FnIO_PrintTERecord( &te, &pPrintH );
                break;
            case SV_BUTTON:
                result = TRUE;
                FnIO_SaveAsTextFile( &te,"\pHelp Text",&vRef,&refNum );
                break;
            default:
                break;



    DisposeHandle( (Handle)te );
    DisposeControl( vScroll );
    DisposeHandle( (Handle)pPrintH );
    DisposDialog( dialog );

    return( result );



/********** PASCAL MyHelpEventFilter */

pascal Boolean MyHelpEventFilter( DialogPtr      theDialog,
                                  EventRecord   *theEvent,
                                  short         *itemHit )

    short           thePart;
    char             key;
    short           itemType;
    Handle           itemHandle;
    Rect             itemRect;
    long             finalTicks;
    Rect             dragRect;
    Boolean          result;
    Point            theLocation;
    WindowPtr        theWindow;

    result = FALSE;
    dragRect = qd.screenBits.bounds;

        switch( (*theEvent).what )
```

```
case mouseDown:
    thePart = FindWindow( (*theEvent).where, &theWindow );
    if( theWindow == theDialog )

        switch( thePart )

            case inDrag:
                DragWindow( theDialog,
                            (*theEvent).where,
                            &dragRect );
                result = TRUE;
                break;
            case inContent:
                if( theDialog == FrontWindow() )

                    MyHelpDoContent( (WindowPtr)theDialog,
                        theEvent, &te, &vScroll, itemHit );
                    result = TRUE;

                break;
            default:
                break;


    break;
case keyDown:
case autoKey:
    key = (*theEvent).message & charCodeMask;
    if( (key == RETURN_KEY) || (key == ENTER_KEY) )

        GetDItem( theDialog,
                  OK_BUTTON,
                  &itemType,
                  &itemHandle,
                  &itemRect );
        HiliteControl( (ControlHandle)itemHandle,
                       inButton );
        Delay( VISUAL_DELAY, &finalTicks );
        HiliteControl( (ControlHandle)itemHandle, 0 );
        *itemHit = OK_BUTTON;
        result = TRUE;

    /* Handle other keyboard equivalents here */
    break;
case updateEvt:
    if( (WindowPtr)(*theEvent).message != theDialog )

        MyDoUpdateWindow( (WindowPtr)(*theEvent).message );

    else

        FnTE_UpdateWindow( theDialog, &te, FALSE );
        FnTE_FrameRect( (WindowPtr)theDialog, &te, 4 );
        FnMisc_FrameButton( theDialog, OK_BUTTON );

    break;
case activateEvt:
    if( (WindowPtr)(*theEvent).message != theDialog )

        /*
        DoActivate_( (WindowPtr)(*theEvent).message,
                     ((*theEvent).modifiers & activeFlag),
```

```
                                        *theEvent );
                        */

                    break;
                default:
                    break;


    return( result );


/********** SetUpHelpTEWindow */

Boolean SetUpHelpTEWindow(
    WindowPtr   w,
    Boolean     hasGrowIcon,
    Boolean     wordWrap,
    short       font,
    short       textSize,
    short       textInsetPixels,
    // results //
    TEHandle        *te,
    ControlHandle *vScroll )
/*
    FnSetUpWindow takes information you supply about a window
    and it's default text parameters then returns a TEHandle to
    a text pane that fills the window and a ControlHandle to
    a scroll bar that is located at the right of the window.
    Returns FALSE if error detected during routine.
*/

    Rect        destRect, viewRect;
    Rect        vSBarRect;
    GrafPtr     oldPort;
    int         SBarWidth = 15;
    int         inset = 10;
    int         bottomInset = 46;

    // define basic te stuff
    GetPort( &oldPort );
    SetPort( w );
    TextFont( font );
    TextSize( textSize );

    // set scroll bar rect
    vSBarRect = (*w).portRect;
    vSBarRect.left = vSBarRect.right - SBarWidth - inset;
    vSBarRect.right = vSBarRect.right + 1 - inset;
    vSBarRect.top = vSBarRect.top - 1 + inset;
    if( hasGrowIcon )
        vSBarRect.bottom = vSBarRect.bottom - 14 - bottomInset;
    else
        vSBarRect.bottom = vSBarRect.bottom - bottomInset;

    // create new control
    *vScroll = NewControl( w, &vSBarRect, "\p", 1, 0, 0, 0,
        scrollBarProc, 0L);
    if( *vScroll == NULL )
        return FALSE;

    // define te view and dest rects
    viewRect = w->portRect;
```

```
    viewRect.right = vSBarRect.left;
    viewRect.bottom = vSBarRect.bottom;
    viewRect.top = vSBarRect.top;
    viewRect.left = viewRect.left + inset;
    InsetRect(&viewRect, textInsetPixels, textInsetPixels);
    destRect = viewRect;

    // create new te record
    *te = TENew( &destRect, &viewRect );
    if( *te == NULL )
        return FALSE;
    if( wordWrap )
        (***te).crOnly = 1;
    else
        (***te).crOnly = -1;

    SetPort( oldPort );
    return TRUE;



/********** MyHelpDoContent */

void MyHelpDoContent( WindowPtr       w,
                      EventRecord     *e,
                      TEHandle        *te,
                      ControlHandle *vScroll,
                      short           *itemHit )
/*
    Use this procedure when user clicks in content region of TE window.
    Uses the TnTE_ScrollProc below to take care of scroll bar function.
    Also uses FnTE_AdjustText function.
*/

    int                 cntlCode;
    ControlHandle       cntl, compareCntl;
    int                 pageSize;
    GrafPtr             savePort;

    GetPort(&savePort);
    SetPort(w);

    *itemHit = 0;

    GlobalToLocal(&e->where);
    if( (cntlCode = FindControl(e->where, w, &cntl)) == inButton )

        if( TrackControl( cntl, e->where, 0L ) )

            if(EqualString((**cntl).contrlTitle,okButtonStr,0,0))
                *itemHit = OK_BUTTON;
            if(EqualString((**cntl).contrlTitle,printButtonStr,0,0))
                *itemHit = PR_BUTTON;
            if(EqualString((**cntl).contrlTitle,saveButtonStr,0,0))
                *itemHit = SV_BUTTON;


    else if( cntlCode == inThumb )

        TrackControl( cntl, e->where, 0L );
        FnTE_AdjustText( te, vScroll );
```

```
    else if( (cntlCode == inUpButton) ||
                (cntlCode == inDownButton) ||
                (cntlCode == inPageUp) ||
                (cntlCode == inPageDown) )

            tempTEHptr = te;
            tempVScrollptr = vScroll;
            TrackControl( cntl, e->where, FnTE_ScrollProc );


    SetPort(savePort);



/*****************************************************************************

    ArgusHelp.r

*****************************************************************************/

#include <Types.r>

resource 'DLOG' (598, "Help Dialog")

45, 10, 330, 405,

movableDBoxProc, /* movableDBoxProc or noGrowDocProc */

invisible,

goAway,

0x0,

598,

"Application Help"
;

resource 'DITL' (598, "Help Dialog")


/* array DITLarray: 3 elements */
            /* [1] */


254, 322, 274, 382,


Button



enabled,



"OK"


,
            /* [2] */
```

254, 249, 274, 309,

Button

enabled,

"Print..."

,
		/* [3] */

254, 176, 274, 236,

Button

enabled,

"Save..."

,

0, 0, 0, 0,

HelpItem

  disabled,

  HMScanhdlg

  598

;

data 'TEXT' (598, "Help Text")

$"4170 706C 6963 6174 696F 6E20 4865 6C70" /* Application Help */

$"2054 6578 742E 2E2E" /* Text... */

```
;


/*****************************************************************

    ArgusAboutBalloons.r

*****************************************************************/

#include <Types.r>
#include <BalloonTypes.r>

resource 'hdlg' (598, "Help Balloons", purgeable)
  /* Header */
  HelpMgrVersion,
  0,                        /* start help with first item in DITL */
  hmSaveBitsNoWindow, /* hmDefaultOptions or hmSaveBitsNoWindow */
  0,                        /* balloon definition */
  0,                        /* variation code or position code, reference
                               IM: More Macintosh Toolbox p3-10 */

  /* Missing Component */
  HMSkipItem
  ,

  /* Help */

    /* [1] */
    HMStringREsItem  /* store help messages in STR# 500 */
     0, 0 ,             /* default tip location is  0, 0  */
     0, 0, 0, 0 ,     /* default alternate rectangle */
    598, 1,                 /* OK button */
    0, 0,                   /* never dimmed */
    0, 0,                   /* never checked */
    0, 0                    /* never marked */
    ,

    /* [2] */
    HMStringREsItem
     0, 0 ,
     0, 0, 0, 0 ,
    598, 2,                 /* Print button */
    0, 0,
    0, 0,
    0, 0
    ,

    /* [3] */
    HMStringREsItem
     0, 0 ,
     0, 0, 0, 0 ,
    598, 3,                 /* Save button */
    0, 0,
    0, 0,
    0, 0
    ,


;

resource 'STR#' (598, "Help Dialog Help Strings")
```

```
    /* [1] */
    "To dismiss dialog, click this button.";
    /* [2] */
    "Use to print above help information.";
    /* [3] */
    "Use to save above help information to a text file.";

;

// End of File
```