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

    Main.cp

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

/*
    Starter Application 2.1

    Requires Argus Libraries 2.1

    Funct_()    All functions and sub-routines in Main.cp are identified
                with an underscore '_' at the end of the function name.
                This hopefully helps to identify program functions
                from Toolbox calls.

    MyFunct()   All other functions are identified by "My" at the
                beginning of the function name.

*/

/********** Standard Includes */
#include <Types.h>
#include <Quickdraw.h>
#include <Controls.h>
#include <Desk.h>
#include <Dialogs.h>
#include <DiskInit.h>
#include <Editions.h>
#include <EPPC.h>
#include <Events.h>
#include <Fonts.h>
#include <GestaltEqu.h>
#include <Lists.h>
#include <Menus.h>
#include <OSEvents.h>
#include <TextEdit.h>
#include <ToolUtils.h>
#include <Traps.h>
#include <AppleEvents.h>
#include <Balloons.h>

#include "Fn_Prototypes.h"
#include "ArgusAbout.h"
#include "ArgusHelp.h"
#include "MainWindow.h"
#include "SampleDialog.h"
#include "SampleList.h"

/********** Defines */
#define BASE_RES                  400
#define NIL_PTR                   0L
#define MOVE_TO_FRONT             -1L
#define LEAVE_IT                  FALSE
#define DRAG_THRESHOLD            30
#define REMOVE_ALL_EVENTS         0
#define MIN_SLEEP                 60L
#define NIL_MOUSE_REGION          0L
#define WNE_TRAP_NUM              0x60
#define UNIMPL_TRAP_NUM           0x9F
#define NIL_STR                   "\p"
#define MIN_SYS_VERSION           0x700   // identify min sys req
```

```
#define HELP_MENU                "\pApplication Help..."

/********** General */
#define ZERO                  0
#define FIRST                 1
#define SECOND                2
#define THIRD                 3
#define FOURTH                4

#define LINE_FEED            10
#define RETURN               13
#define SPACE                32
#define TAB                   9

/********** Window definitions */
#define WIND_LEFT            10
#define WIND_TOP             50
#define WIND_OFFSET          10

/*
     The following define statements cross-reference
     names with resource ID numbers.
*/

/********** Menus */
#define APPLE_MENU        BASE_RES
#define FILE_MENU         BASE_RES + 1
#define EDIT_MENU         BASE_RES + 2

/********** Error Strings */
#define GENERAL_ERR       900
#define BAD_SYS           901
#define N0_RESOURCE       902

/********** Apple Menu Stuff */
#define ABOUT             1

/********** File Menu Stuff */
#define NEW               1
#define CLOSE             2
#define QUIT              3

/********** Edit Menu Stuff */
#define UNDO              1
#define CUT               3
#define COPY              4
#define PASTE             5
#define CLEAR             6

/********** Structures */
struct SysConfigRec

     Boolean hasGestalt;
     Boolean hasWNE;
     Boolean hasColorQD;
     Boolean hasAppleEvents;
     Boolean hasEditionMgr;
     Boolean hasHelpMgr;

     long      sysVersion;
;
```

```
/********** Variables */
Boolean                 gDone;
EventRecord             gTheEvent;
MenuHandle              gAppleMenu;
MenuHandle              gFileMenu;
MenuHandle              gEditMenu;
Rect                    gDragRect;
short                   gAppResourceFile;
struct SysConfigRec gSysConfig;
int                     gNewWindowLeft = WIND_LEFT;
int                     gNewWindowTop = WIND_TOP;

/********** Prototypes */
void main( void );
void ToolBoxInit_( void );
Boolean TrapAvailable_( short tNumber, TrapType tType );
static void GetSysConfig_( void );
void MenuBarInit_( void );
void SetUpDragRect_( void );
void MainLoop_( void );
void HandleEvent_( void );
void HandleMouseDown_( void );
void AdjustMenus_( void );
int   IsDAWindow_( WindowPtr whichWindow );
void HandleMenuChoice_( long int menuChoice );
void HandleAppleChoice_( int theItem );
void HandleFileChoice_( int theItem );
void HandleEditChoice_( int theItem );
void NewWindow_( void );
void Quit_( void );

/********** main */

void main( void )

    ToolBoxInit_();
    GetSysConfig_();

    if( gSysConfig.hasAppleEvents )
        FnAE_InitAE();

    MenuBarInit_();
    SetUpDragRect_();

    if( !gSysConfig.hasAppleEvents )
        /* put OpenApp procedure here */
        MyCreateWindow( BASE_RES, NIL_PTR, (WindowPtr)MOVE_TO_FRONT,
            WIND_TOP, WIND_LEFT, WIND_OFFSET, N0_RESOURCE );

    MainLoop_();



/********** ToolBoxInit */

void ToolBoxInit_( void )
/*
    Standard initialization procedure per IM:Overview p4-75
*/

    MaxApplZone();
    MoreMasters();
```

```
	InitGraf( &qd.thePort );
	InitFonts();
	InitWindows();
	InitMenus();
	TEInit();
	InitDialogs( NIL_PTR );

	FlushEvents( everyEvent, REMOVE_ALL_EVENTS );
	InitCursor();



/********** TrapAvailable */

Boolean TrapAvailable_( short tNumber, TrapType tType )

	return( NGetTrapAddress( tNumber, tType )
	!= GetTrapAddress( _Unimplemented ) );



/********** GetSysConfig */

static void GetSysConfig_( void )

	OSErr		ignoreError;
	long		tempLong;
	SysEnvRec	environs;
	short		myBit;


	gAppResourceFile = CurResFile();	/* set app resource fork ID */

	/* Check to see if Gestalt Manager is supported */
	gSysConfig.hasGestalt = TrapAvailable_( _Gestalt, ToolTrap );
	if( !gSysConfig.hasGestalt )
		/* Something has got to be wrong */
		FnErr_DisplayStrID( BAD_SYS, TRUE );
	else

		/* Determine various system configuration checks */

		/* Check to see if WaitNextEvent is supported */
		gSysConfig.hasWNE = TrapAvailable_( _WaitNextEvent, ToolTrap );

		/* Check for Color Capability */
		ignoreError = Gestalt( gestaltQuickdrawVersion, &tempLong );
		gSysConfig.hasColorQD = ( tempLong != gestaltOriginalQD );

		/* Check to see if AppleEvents are supported in OS */
		gSysConfig.hasAppleEvents = ( Gestalt( gestaltAppleEventsAttr,
			&tempLong ) == noErr );

		/* Check for Edition Manager */
		gSysConfig.hasEditionMgr = ( Gestalt( gestaltEditionMgrAttr,
			&tempLong ) == noErr );
		if( gSysConfig.hasEditionMgr )
			if( InitEditionPack() != noErr )
				gSysConfig.hasEditionMgr = false;

		/* Check for Help Manager */
```

```
        ignoreError = Gestalt( gestaltHelpMgrAttr, &tempLong );
        myBit = gestaltHelpMgrPresent;
        gSysConfig.hasHelpMgr =
            BitTst( &tempLong, 31 - myBit );

        /* Determine and Check OS Version */
        gSysConfig.sysVersion = 0.0;
        ignoreError = Gestalt( gestaltSystemVersion, &tempLong );
        gSysConfig.sysVersion = tempLong;
        if( MIN_SYS_VERSION > gSysConfig.sysVersion )
            FnErr_DisplayStrID( BAD_SYS, TRUE );



/********** MenuBarInit      */

void MenuBarInit_( void )

    Handle      myMenuBar;
    MenuHandle helpMenu;
    OSErr       err;

    myMenuBar = GetNewMBar( BASE_RES ) ;
    if ( myMenuBar == NIL_PTR )
        FnErr_DisplayStrID( N0_RESOURCE, TRUE );
    SetMenuBar( myMenuBar );

    if ( ( gAppleMenu = GetMHandle( APPLE_MENU ) ) == NIL_PTR )
        FnErr_DisplayStrID( N0_RESOURCE, TRUE );
    if ( ( gFileMenu = GetMHandle( FILE_MENU ) ) == NIL_PTR )
        FnErr_DisplayStrID( N0_RESOURCE, TRUE );
    if ( ( gEditMenu = GetMHandle( EDIT_MENU ) ) == NIL_PTR )
        FnErr_DisplayStrID( N0_RESOURCE, TRUE );

    AddResMenu( gAppleMenu, 'DRVR' );
    DrawMenuBar();

    if( gSysConfig.hasHelpMgr )

        err = HMGetHelpMenuHandle( &helpMenu );
        if( err == noErr )

            if( helpMenu != nil )

                AppendMenu( helpMenu, HELP_MENU );




/********** SetUpDragRect */

void SetUpDragRect_( void )

    gDragRect = qd.screenBits.bounds;
    gDragRect.left += DRAG_THRESHOLD;
    gDragRect.right -= DRAG_THRESHOLD;
    gDragRect.bottom -= DRAG_THRESHOLD;
```

```
/********** MainLoop */

void MainLoop_( void )

    gDone = FALSE;
    while( gDone == FALSE )

        HandleEvent_();



/********** HandleEvent */

void HandleEvent_( void )

    char      theChar;
    GrafPtr oldPort;
    Point     displayPoint;

    if ( gSysConfig.hasWNE )
        WaitNextEvent( everyEvent, &gTheEvent, MIN_SLEEP,
            NIL_MOUSE_REGION );
    else

        SystemTask();
        GetNextEvent( everyEvent, &gTheEvent );


    switch ( gTheEvent.what )

        case mouseDown:
            HandleMouseDown_();
            break;
        case mouseUp:
            // this application doesn't use mouseUp events
            break;
        case keyDown:
        case autoKey:
            theChar = gTheEvent.message & charCodeMask;
            if (( gTheEvent.modifiers & cmdKey ) != 0)

                AdjustMenus_();
                HandleMenuChoice_( MenuKey( theChar ) );

            else

                /* Handle text from keyboard */

            break;
        case updateEvt:
            if ( !IsDAWindow_( (WindowPtr)gTheEvent.message ) )

                /* Handle update event */
                MyDoUpdateWindow( (WindowPtr)gTheEvent.message );

            break;
        case diskEvt:
            // most applications don't need to worry about diskEvt's
              break;
        case activateEvt:
```

```
            if ( !IsDAWindow_( (WindowPtr)gTheEvent.message ) )

                if ( gTheEvent.modifiers & activeFlag )

                    /* Handle activate event. */
                    MyDoActivateWindow((WindowPtr)gTheEvent.message);

                else

                    /* Handle deactivate event. */
                    MyDoDeactivateWindow((WindowPtr)gTheEvent.message);


            break;
        case osEvt:
            // this application doesn't support operating sys events
            break;
        case nullEvent:
            // ignore
            break;
        case kHighLevelEvent:
            // need to #include <Events.h> to define kHighLevelEvent
            FnAE_DoHighLevelEvent( &gTheEvent );
            break;



/********** HandleMouseDown */

void HandleMouseDown_( void )

    WindowPtr     whichWindow;
    short int     thePart;
    long int      menuChoice;
    Point         theLocation;

    thePart = FindWindow( gTheEvent.where, &whichWindow );
    switch ( thePart )

        case inMenuBar:
            AdjustMenus_();
            menuChoice = MenuSelect( gTheEvent.where );
            HandleMenuChoice_( menuChoice );
            break;
        case inSysWindow:
            SystemClick( &gTheEvent, whichWindow );
            break;
        case inContent:
            if( whichWindow != FrontWindow() )
                SelectWindow( whichWindow );
            else if( !IsDAWindow_( (WindowPtr)gTheEvent.message ) )
                /* Handle click in window content */
                MyDoContent( whichWindow, &gTheEvent );
            break;
        case inDrag:
            if( whichWindow != FrontWindow() )

                SelectWindow( whichWindow );
                DragWindow( whichWindow, gTheEvent.where, &gDragRect );

            else
```

```
                DragWindow( whichWindow, gTheEvent.where, &gDragRect );
            break;
        case inGrow:
            // not used in this application
            break;
        case inGoAway:
            theLocation = gTheEvent.where;
            GlobalToLocal( &theLocation );
            if( TrackGoAway( whichWindow, theLocation ) )
                DisposeWindow( whichWindow );
            break;
        case inZoomIn:
        case inZoomOut:
            // not used in this application
            break;



/********** AdjustMenus */

void AdjustMenus_( void )

    WindowPtr currentWindow;

    if (IsDAWindow_( FrontWindow() ) )

        EnableItem( gEditMenu, UNDO );
        EnableItem( gEditMenu, CUT );
        EnableItem( gEditMenu, COPY );
        EnableItem( gEditMenu, PASTE );
        EnableItem( gEditMenu, CLEAR );

    else

        DisableItem( gEditMenu, UNDO );
        DisableItem( gEditMenu, CUT );
        DisableItem( gEditMenu, COPY );
        DisableItem( gEditMenu, PASTE );
        DisableItem( gEditMenu, CLEAR );


    if ( ( currentWindow = FrontWindow() ) == NIL_PTR )
        DisableItem( gFileMenu, CLOSE );
    else
        EnableItem( gFileMenu, CLOSE );



/********** IsDAWindow */

int IsDAWindow_( WindowPtr whichWindow )

    if ( whichWindow == NIL_PTR )
        return( FALSE );
    else /* DA windows have negative windowKinds */
        return( ( (WindowPeek)whichWindow )->windowKind < 0 );



/********** HandleMenuChoice */
```

```
void HandleMenuChoice_( long int menuChoice )

    int     theMenu;
    int     theItem;

    if ( menuChoice != 0 )

        theMenu = HiWord( menuChoice );
        theItem = LoWord( menuChoice );
        switch ( theMenu )

            case APPLE_MENU :
                HandleAppleChoice_( theItem );
                break;
            case FILE_MENU :
                HandleFileChoice_( theItem );
                break;
            case EDIT_MENU :
                HandleEditChoice_( theItem );
                break;
            case kHMHelpMenuID :
                MyHelpDialog();
                break;

        HiliteMenu( 0 );



/********** HandleAppleChoice */

void HandleAppleChoice_( int theItem )

    Str255   accName;
    int      accNumber;

    switch ( theItem )

        case ABOUT :
            MyAboutDialog();
            break;
        default :
            GetItem( gAppleMenu, theItem, accName );
            accNumber = OpenDeskAcc( accName );
            break;



/********** HandleFileChoice */

void HandleFileChoice_( int theItem )

    WindowPtr    whichWindow;

    switch ( theItem )

        case NEW :
            NewWindow_();
            break;
        case CLOSE :
            if ( ( whichWindow = FrontWindow() ) != NIL_PTR )
```

```
                DisposeWindow( whichWindow );
            break;
        case QUIT :
            Quit_();
            break;



/********** HandleEditChoice */

void HandleEditChoice_( int theItem )

    if( SystemEdit( theItem - 1 ) == 0 )

        /* Add Edit menu switch statement here */



/********** NewWindow */

void NewWindow_( void )

    MyCreateWindow( BASE_RES, NIL_PTR, (WindowPtr)MOVE_TO_FRONT,
        WIND_TOP, WIND_LEFT, WIND_OFFSET, N0_RESOURCE );



/********** Quit */

void Quit_( void )

    gDone = TRUE;



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

    Main.r

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

#include <Types.r>

resource 'MBAR' (400, "Main MBAR")


/* array MenuArray: 3 elements */


/* [1] */


400,


/* [2] */


401,
```

```
/* [3] */

402

;
resource 'MENU' (400, "Apple")
400,
textMenuProc,
0x7FFFFFFD,
enabled,
apple,

/* array: 2 elements */

/* [1] */

"About Application...", noIcon, noKey, noMark, plain,

/* [2] */

"-", noIcon, noKey, noMark, 1

;
resource 'MENU' (401, "File")
401,
textMenuProc,
allEnabled,
enabled,
"File",

/* array: 3 elements */

/* [1] */

"New", noIcon, "N", noMark, plain,

/* [2] */
```

"Close", noIcon, "W", noMark, plain,

/* [3] */

"Quit", noIcon, "Q", noMark, plain

;

resource 'MENU' (402, "Edit", preload)

402,

textMenuProc,

0x0,

enabled,

"Edit",

/* array: 6 elements */

/* [1] */

"Undo", noIcon, "Z", noMark, plain,

/* [2] */

"-", noIcon, noKey, noMark, plain,

/* [3] */

"Cut", noIcon, "X", noMark, plain,

/* [4] */

"Copy", noIcon, "C", noMark, plain,

/* [5] */

"Paste", noIcon, "V", noMark, plain,

/* [6] */

```
"Clear", noIcon, noKey, noMark, plain


;


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

      MainBalloons.r

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

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

resource 'hmnu' (400, "Apple", purgeable)
/* header component */
   HelpMgrVersion,      /* version of Help Manager */
   hmDefaultOptions,    /* options */
   0,                   /* balloon definition function */
   0,                   /* variation code */

/* missing-items component */
   HMSkipItem
      /* no missing items, so skip to menu-title component */
      ,

/* BEGIN */

/* menu-title component */
   HMSkipItem
      /* Apple menu has default help balloons */
      ,

/* first menu-item component: About */
   HMStringItem
      /* enabled */
      "Displays information about this application.",
      /* dimmed by application */
      "",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* END */
;

resource 'hmnu' (401, "File", purgeable)
/* header component */
   HelpMgrVersion,      /* version of Help Manager */
   hmDefaultOptions,    /* options */
   0,                   /* balloon definition function */
   0,                   /* variation code */

/* missing-items component */
   HMSkipItem
      /* no missing items, so skip to menu-title component */
      ,

/* BEGIN */
```

```
/* menu-title component */
   HMStringItem
      /* enabled */
      "File menu\n\nUse this menu to open documents or quit "
      "this application.",
      /* dimmed by application */
      "",
      /* dimmed by system (alerts or modal dialogs) */
      "File menu\n\nUse this menu to open documents or quit "
      "this application. Not available because there is a dialog box "
      "on the screen.",
      /* all items dimmed by system */
      "This item is not available because there is a dialog "
      "box on the screen.",
      ,

/* first menu-item component: New */
   HMStringItem
      /* enabled */
      "Creates new document window.",
      /* dimmed by application */
      "",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* second menu-item component: Close */
   HMStringItem
      /* enabled */
      "Closes current document window.",
      /* dimmed by application */
      "Closes current document window. Not available now "
      "because the application does not currently have an "
      "open window.",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* third menu-item component: Quit */
   HMStringItem
      /* enabled */
      "Quits this application.",
      /* dimmed by application */
      "",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* END */
;

resource 'hmnu' (402, "Edit", purgeable)
/* header component */
   HelpMgrVersion,      /* version of Help Manager */
   hmDefaultOptions,    /* options */
```

```
    0,                  /* balloon definition function */
    0,                  /* variation code */

/* missing-items component */
    HMSkipItem
        /* no missing items, so skip to menu-title component */
        ,

/* BEGIN */

/* menu-title component */
    HMStringItem
        /* enabled */
        "Edit menu\n\nUse this menu to manipulate text.",
        /* dimmed by application */
        "Edit menu\n\nUse this menu to manipulate text. "
        "Not available because there is nothing to edit.",
        /* dimmed by system (alerts or modal dialogs) */
        "Edit menu\n\nUse this menu to manipulate text. "
        "Not available because there is a dialog box on "
        "the screen.",
        /* all items dimmed by system */
        "This item is not available because there is a dialog "
        "box on the screen.",
        ,

/* first menu-item component: Undo */
    HMStringItem
        /* enabled */
        "Cancels your last edit.",
        /* dimmed by application */
        "Cancels your last edit. Not available because "
        "application doesn't support this function.",
        /* checked */
        "",
        /* marked */
        "",
        ,

/* second menu-item component: Divider */
    HMSkipItem
        /* no help balloons for divider lines */
        ,

/* third menu-item component: Cut */
    HMStringItem
        /* enabled */
        "Cuts selected text to the Clipboard.",
        /* dimmed by application */
        "Cuts selected text to the Clipboard. Not available now "
        "because no text is selected.",
        /* checked */
        "",
        /* marked */
        "",
        ,

/* forth menu-item component: Copy */
    HMStringItem
        /* enabled */
        "Copies selected text to the Clipboard.",
        /* dimmed by application */
```

```
      "Copies selected text to the Clipboard. Not available now "
      "because no text is selected.",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* fifth menu-item component: Paste */
   HMStringItem
      /* enabled */
      "Inserts selected text from the Clipboard.",
      /* dimmed by application */
      "Inserts selected text from the Clipboard. Not available "
      "now because there is no text currently in the Clipboard.",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* sixth menu-item component: Clear */
   HMStringItem
      /* enabled */
      "Deletes selected text.",
      /* dimmed by application */
      "Deletes selected text. Not available now "
      "because no text is selected.",
      /* checked */
      "",
      /* marked */
      "",
      ,

/* END */
;

resource 'hmnu' (kHMHelpMenuID, "Help", purgeable)

      HelpMgrVersion, 0, 0, 0,   /* header component                  */
      HMSkipItem                 /* missing-items component           */
            /* no missing items, skip to first appended menu-item */
            /* component                                             */
            ,
           /* first menu-item component */
            HMStringResItem   /* use an 'STR#' for help messages   */
            800, 1,   /* 'STR#' res ID, index when item is enabled */
            800, 2,   /* 'STR#' res ID, index when item is dimmed   */
            800, 3,   /* 'STR#' res ID, index when item is checked */
            0, 0       /* item cannot be marked */
            ,

;

resource 'STR#' (800, "Help menu item strings")


            /* array StringArray: six elements */
            /* [1] enabled help command */
            "Provides help for this application.";
            /* [2] dimmed help command */
            "Provides help for this application.";
```

```
        /* [3] checked help command */
        "Provides help for this application.";

;

// End of File
```