PROGRAMMING THE MAC : about, compiler, references, sample application

---

## ABOUT THIS SECTION

This section does NOT provide a detailed explanation on how to program for the Macintosh. There are other (and better) sources for this type of information. What this section DOES provide is a suggested approach for learning how to start programming for the Macintosh and what sort of tools and documentation you'll need based on personal experience. Also provided is a sample application which can be used as a reference and a starting point for your own applications. Learning C or C++ is a fairly straightforward experience. Attempting the tackle the Toolbox (the function calls which make up the MacOS) can be a much more daunting task. Try to be patient and don't expect to learn everything in one day.

You'd think that learning to program the Macintosh (or any graphical environment) would consist of buying one book, start at the beginning, and a few days later finish the book with a thorough knowledge of everything you needed to know to master the Toolbox. Well, it simply ain't so. There are literally volumes of reference material from Apple on all of the Managers which comprise the MacOS. Managers are groups of related functions. Examples of which are the QuickDraw Manager, TextEdit Manager, Menu Manager, etc. There are dozens of Managers and THOUSANDS of functions which comprise the Toolbox. Very few people can expect to master them all.

## PURCHASING A COMPILER

There are basically two choices for C/C++ compilers for the Macintosh; Think C by Symantec and CodeWarrior by Metrowerks. Both are similarly priced and both are full-featured compilers. Think C has been around for longer, but Code Warrior has made major inroads since it's introduction. Either choice is money well spent.

## REFERENCE MATERIAL

The following is the minimum recommended reference material you should start out with if you intend on programming for the Macintosh:

Inside Macintosh: Overview
Inside Macintosh: Macintosh Toolbox Essentials
Inside Macintosh: Imaging with QuickDraw
Symantec THINK Reference and/or APDA Toolbox Assistant
A good C/C++ book

There are currently over two dozen "Inside Macintosh" books available from Apple. You can easily order these books (as well as your compiler) through APDA (Apple Program Developers Association). While your at it, consider becoming a member of APDA. APDA can be reached at 1-800-282-2732 or over the net at http://www.info.apple.com/dev.

## A COMPLETE SAMPLE APPLICATION

The following is a complete application. It represents watered-down code in that there is limited error-checking and no support for things like AppleEvents, Scripting, or Balloon Help. However, it does provide the basic event-loop which almost all Macintosh applications are based on.

The sample application consists of two files; starter.cp and resources.r

/************************ Segment: starter.cp *************************

All functions and sub-routines are identified with an underscore '_'
at the end of the function name. This hopefully helps to identify

```
   program functions from Toolbox calls.

*/

/********** 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>

/********** Define Statements */
#define kNilPtr                  0L
#define kMoveToFront             (WindowPtr)-1L
#define kLeaveIt                 false
#define kRemoveEvents            0
#define kSleep                   60L
#define kNilMouseRegion          0L
#define kWaitNextEventTrap   0x60
#define kUnimplementedTrap   0x9F
#define kMinimumSysVersion   0x700   // minimum system version required

/********** General */
const short kDragThreshold = 30;
const unsigned char *kNilStr = "\p";
const unsigned char *kFatalErrorStr = "\pFatal Error!";

/********** Window definitions */
const short kNewWindow =      400;
const short kWindowLeft =     5;
const short kWindowTop =      45;
const short kWindowOffset = 20;

/********** Alerts */
const short kAboutAlert =     400;
const short kErrorAlert =     401;

/********** Param Text */
const short kGeneralError = 400;
const short kBadSystem =      401;
const short kNoMenuBarRes = 402;
const short kNoMenuRes =      403;
const short kNoWindowRes =    404;

/********** Menu Stuff */
const short kMenuBar =          400;
const short kAppleMenu =      400;
const short kFileMenu =         401;
```

```
const short kEditMenu =        402;
const short kAppleAbout =     1;
const short kFileNew =         1;
const short kFileClose =      2;
const short kFileQuit =       3;
const short kEditUndo =        1;
const short kEditCut =         2;
const short kEditCopy =        3;
const short kEditPaste =      4;
const short kEditClear =      6;

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

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

      long      sysVersion;
;

/********** Global Variables */
Boolean                   gDone;
EventRecord             gTheEvent;
MenuHandle               gAppleMenu;
MenuHandle               gFileMenu;
MenuHandle               gEditMenu;
Rect                       gDragRect;
short                     gAppResourceFile;
struct SysConfigRec gSysConfig;
int                       gNewWindowLeft = kWindowLeft;
int                       gNewWindowTop = kWindowTop;

/********** Prototypes */
void main( void );
void ToolBoxInit_( void );
static void GetSysConfig_( void );
Boolean TrapAvailable_( short tNumber, TrapType tType );
void MenuBarInit_( void );
void SetUpDragRect_( void );
void HandleEvent_( void );
void DoMouseDown_( void );
void AdjustMenus_( void );
short IsDAWindow_( WindowPtr w );
void DoMenuChoice_( long int menuChoice );
void DoAppleChoice_( int theItem );
void DoFileChoice_( int theItem );
void DoEditChoice_( int theItem );
void DoCreateWindow_( void );
void ErrorHandler_( int stringNum, int quitFlag );


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

void main( void )

      ToolBoxInit_();
      GetSysConfig_();
      MenuBarInit_();
```

```
	SetUpDragRect_();
	DoCreateWindow_();

	gDone = false;
	while( gDone == false )
		HandleEvent_();



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

void ToolBoxInit_( void )

	// Standard initialization procedure per IM:Overview p4-75

	MaxApplZone();
	MoreMasters();

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

	FlushEvents( everyEvent, kRemoveEvents );
	InitCursor();



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

static void GetSysConfig_( void )

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

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

	// check to see if Gestalt Manager is supported
	gSysConfig.hasGestalt = TrapAvailable_( _Gestalt, ToolTrap );
	if( !gSysConfig.hasGestalt )
		// something has got to be wrong
		ErrorHandler_( kGeneralError, true );
	else

		// determine system configuration

		gSysConfig.hasWNE = TrapAvailable_( _WaitNextEvent, ToolTrap );

		ignoreError = Gestalt( gestaltQuickdrawVersion, &tempLong );
		gSysConfig.hasColorQD = ( tempLong != gestaltOriginalQD );

		gSysConfig.hasAppleEvents = ( Gestalt( gestaltAppleEventsAttr,
			&tempLong ) == noErr );

		gSysConfig.hasEditionMgr = ( Gestalt( gestaltEditionMgrAttr,
			&tempLong ) == noErr );
		if( gSysConfig.hasEditionMgr )
```

```
            if( InitEditionPack() != noErr )
                gSysConfig.hasEditionMgr = false;

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

        gSysConfig.sysVersion = 0.0;
        ignoreError = Gestalt( gestaltSystemVersion, &tempLong );
        gSysConfig.sysVersion = tempLong;
        if( kMinimumSysVersion > gSysConfig.sysVersion )
            ErrorHandler_( kBadSystem, true );



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

Boolean TrapAvailable_( short tNumber, TrapType tType )

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



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

void MenuBarInit_( void )

    Handle   myMenuBar;

    if( ( myMenuBar = GetNewMBar( kMenuBar ) ) == kNilPtr )
        ErrorHandler_( kNoMenuBarRes, true );
    SetMenuBar( myMenuBar );

    gAppleMenu = GetMHandle( kAppleMenu );
    gFileMenu = GetMHandle( kFileMenu );
    gEditMenu = GetMHandle( kEditMenu );
    if( gAppleMenu == kNilPtr || gFileMenu == kNilPtr ||
      gEditMenu == kNilPtr ) ErrorHandler_( kNoMenuRes, true );
    AddResMenu( gAppleMenu, 'DRVR' );
    DrawMenuBar();



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

void SetUpDragRect_( void )

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



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

void HandleEvent_( void )

    char        theChar;
```

```
GrafPtr     oldPort;
WindowPtr w;

if( gSysConfig.hasWNE )
     WaitNextEvent( everyEvent, &gTheEvent, kSleep, kNilMouseRegion );
else

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


switch( gTheEvent.what )

     case mouseDown:
          DoMouseDown_();
          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_();
               DoMenuChoice_( MenuKey( theChar ) );

          else

               /* Handle text from keyboard */

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

               w = (WindowPtr)gTheEvent.message;
               GetPort( &oldPort );
               SetPort( w );
               BeginUpdate( w );

                    // perform and update/drawing here

               EndUpdate( w );
               SetPort( oldPort );

          break;
     case diskEvt:
          // most applications don't need to worry about diskEvt's
            break;
     case activateEvt:
          // this application doesn't need to worry about activateEvt's
          if( !IsDAWindow_( (WindowPtr)gTheEvent.message ) )

               if( gTheEvent.modifiers & activeFlag )

                    /* Handle activate event. */

               else

                    /* Handle deactivate event. */
```

```
            break;
        case osEvt:
            // this application doesn't support operating sys events
            break;
        case nullEvent:
            // ignore
            break;
    /*
        case kHighLevelEvent:
            // this application doesn't support high level events
            // need to #include <Events.h> to define kHighLevelEvent
            break;
    */



/********** DoMouseDown */

void DoMouseDown_( void )

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

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

        case inMenuBar:
            AdjustMenus_();
            menuChoice = MenuSelect( gTheEvent.where );
            DoMenuChoice_( menuChoice );
            break;
        case inSysWindow:
            SystemClick( &gTheEvent, w );
            break;
        case inContent:
            if( w != FrontWindow() )
                SelectWindow( w );
            else if( !IsDAWindow_( (WindowPtr)gTheEvent.message ) )
                /* Handle click in window content */ ;
            break;
        case inDrag:
            DragWindow( w, gTheEvent.where, &gDragRect );
            break;
        case inGrow:
            // not used in this application
            break;
        case inGoAway:
            theLocation = gTheEvent.where;
            GlobalToLocal( &theLocation );
            if( TrackGoAway( w, theLocation ) )
                DisposeWindow( w );
            break;
        case inZoomIn:
        case inZoomOut:
            // not used in this application
            break;
```

```
/********** AdjustMenus */

void AdjustMenus_( void )

    WindowPtr w;

    if(IsDAWindow_( FrontWindow() ) )

        EnableItem( gEditMenu, kEditUndo );
        EnableItem( gEditMenu, kEditCut );
        EnableItem( gEditMenu, kEditCopy );
        EnableItem( gEditMenu, kEditPaste );
        EnableItem( gEditMenu, kEditClear );

    else

        DisableItem( gEditMenu, kEditUndo );
        DisableItem( gEditMenu, kEditCut );
        DisableItem( gEditMenu, kEditCopy );
        DisableItem( gEditMenu, kEditPaste );
        DisableItem( gEditMenu, kEditClear );


    if( ( w = FrontWindow() ) == kNilPtr )
        DisableItem( gFileMenu, kFileClose );
    else
        EnableItem( gFileMenu, kFileClose );



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

short IsDAWindow_( WindowPtr w )

    if( w == kNilPtr )
        return( false );
    else /* DA windows have negative windowKinds */
        return( ( (WindowPeek)w )->windowKind < 0 );



/********** DoMenuChoice */

void DoMenuChoice_( long int menuChoice )

    int     theMenu;
    int     theItem;

    if( menuChoice != 0 )

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

            case kAppleMenu :
                DoAppleChoice_( theItem );
                break;
            case kFileMenu :
                DoFileChoice_( theItem );
                break;
            case kEditMenu :
```

```
                DoEditChoice_( theItem );
                break;

        HiliteMenu( 0 );



/********** DoAppleChoice */

void DoAppleChoice_( int theItem )

    Str255   accName;
    int      accNumber;

    switch( theItem )

        case kAppleAbout :
            NoteAlert( kAboutAlert, kNilPtr );
            break;
        default :
            GetItem( gAppleMenu, theItem, accName );
            accNumber = OpenDeskAcc( accName );
            break;



/********** DoFileChoice */

void DoFileChoice_( int theItem )

    WindowPtr    w;

    switch( theItem )

        case kFileNew :
            DoCreateWindow_();
            break;
        case kFileClose :
            if( ( w = FrontWindow() ) != kNilPtr )
                DisposeWindow( w );
            break;
        case kFileQuit :
            gDone = TRUE;
            break;



/********** DoEditChoice */

void DoEditChoice_( int theItem )

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

        /* Add Edit menu switch statement here */



/********** DoCreateWindow */
```

```
void DoCreateWindow_( void )

    WindowPtr      w;

    w = GetNewWindow( kNewWindow, kNilPtr, kMoveToFront );
    if(((qd.screenBits.bounds.right - gNewWindowLeft) < kDragThreshold) ||
        ((qd.screenBits.bounds.bottom - gNewWindowTop) < kDragThreshold))

        gNewWindowLeft = kWindowLeft;
        gNewWindowTop = kWindowTop;

    MoveWindow( w, gNewWindowLeft, gNewWindowTop, kLeaveIt );
    gNewWindowLeft += kWindowOffset;
    gNewWindowTop += kWindowOffset;
    ShowWindow( w );



/********** ErrorHandler */

void ErrorHandler_( int stringNum, int quitFlag )

    StringHandle    errorStringH;

    if( ( errorStringH = GetString( stringNum ) ) == kNilPtr )
        ParamText( kFatalErrorStr, kNilStr, kNilStr, kNilStr );
    else

        HLock( (Handle)errorStringH );
        ParamText( *errorStringH, kNilStr, kNilStr, kNilStr );
        HUnlock( (Handle)errorStringH );

    StopAlert( kErrorAlert, kNilPtr );
    if( quitFlag )
        ExitToShell();



/************************ Segment: resources.r ************************/

#include <Types.r>

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

45, 5, 205, 181,

noGrowDocProc,

invisible,

goAway,

0x0,

"Window"
;

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' (400, "Apple")

400,

textMenuProc,

0x7FFFFFFD,

enabled,

apple,

/* array: 2 elements */

/* [1] */

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

/* [2] */

"-", noIcon, noKey, noMark, 1

;
```

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

;

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

/* array MenuArray: 3 elements */

/* [1] */

400,

/* [2] */

401,

/* [3] */

402

;

resource 'DITL' (400, "About")

/* array DITLarray: 2 elements */

/* [1] */

71, 117, 91, 177,

Button

enabled,

"OK"

,

/* [2] */

7, 70, 61, 280,

StaticText

enabled,

"Starter Application - a place to start y"

"our Macintosh program."

;

resource 'DITL' (401, "Fatal Error")

/* array DITLarray: 2 elements */

/* [1] */

86, 117, 106, 177,

Button

enabled,

"Exit"

,

/* [2] */

5, 67, 71, 283,

StaticText

enabled,

"Fatal error:  ^0"

;

resource 'ALRT' (400, "About")

40, 40, 142, 332,

400,

/* array: 4 elements */

/* [1] */

OK, visible, silent,

/* [2] */

OK, visible, silent,

/* [3] */

OK, visible, silent,

/* [4] */

OK, visible, silent

;

resource 'ALRT' (401, "Fatal Error")

40, 40, 156, 332,

401,

/* array: 4 elements */

/* [1] */

OK, visible, sound1,

/* [2] */

OK, visible, sound1,

/* [3] */

OK, visible, sound1,

/* [4] */

OK, visible, sound1

```
;

resource 'STR ' (400, "General Error")

"An unrecoverable error occured!"
;

resource 'STR ' (401, "System Error")

"Need to upgrade to System 7!"
;

resource 'STR ' (402, "MENUBAR Error")

"Couldn't load the MENU Bar resource!"
;

resource 'STR ' (403, "MENU Error")

"Couldn't load a MENU resource!"
;

resource 'STR ' (404, "WIND Error")

"Couldn't load a WIND resource!"
;

// End of File
```