```cpp
/*
  File:   ReversePages.cpp

  Contains: Reverses pages from front to back within a PDF file.

  Written by: Mark Gavin
              Appligent, Inc.
              60 South Lansdowne Avenue
              Lansdowne, PA 19050
              ( 610 ) 284-4006

  Copyright:  ©1997-2005 by Appligent, Inc.

*/

#include "CorCalls.h"
#include "AVCalls.h"
#include "PDCalls.h"
#include "CosCalls.h"
#include "ASCalls.h"

// -------------------------
//
// Utility functions
//
// -------------------------
// display a simple alert dialog with the error message.

void DisplayErrorAlert( ASInt32 inError, const char * inString )
  {
    char          theStringBuffer[128] ;

    ASGetErrorString( inError, theStringBuffer, sizeof( theStringBuffer ) ) ;

    strcat( theStringBuffer, " " ) ;
    strcat( theStringBuffer, inString ) ;

    AVAlert( ALERT_STOP, theStringBuffer, "OK", ( const char * )NULL, ( const char * )NULL, false ) ;

    return ;

  } // end DisplayErrorAlert

// -------------------------
// Routine to handle the majority of menu items.
// Note: this does not allow for the setting of command or accelerator keys.
// inTitle is what is displayed in the user interface.
// inMenuItemName is used internally and is independent of language.

AVMenuItem CreateMenuItem( AVMenu inAVMenu, const char * inTitle, const char * inMenuItemName,
                                      AVComputeEnabledProc inAVComputeEnabledProc, void * inAVComputeEnabledProcData,
                                      AVExecuteProc inAVExecuteProc, void * inAVExecuteProcData )
  {
    AVMenuItem  theAVMenuItem ;

    DURING

      theAVMenuItem = AVMenuItemNew( inTitle, inMenuItemName, ( AVMenu )NULL, false, NO_SHORTCUT, ( ASInt16 )0, NULL, gExtensionID ) ;

      if ( inAVComputeEnabledProc != NULL )
        AVMenuItemSetComputeEnabledProc( theAVMenuItem,
                            ASCallbackCreateProto( AVComputeEnabledProc, inAVComputeEnabledProc ), inAVComputeEnabledProcData ) ;

      if ( inAVExecuteProc != NULL )
        AVMenuItemSetExecuteProc( theAVMenuItem, ASCallbackCreateProto( AVExecuteProc, inAVExecuteProc ), inAVExecuteProcData ) ;

    HANDLER
      DisplayErrorAlert( ERRORCODE, "Error creating new menu item" ) ;
      return ( AVMenuItem )NULL ;
    END_HANDLER

    return theAVMenuItem ;

  } // end CreateMenuItem

// -------------------------
// Add a newly created menu item to a menu following the given menu item.

void AddAfterMenuItem( const char *inTitle, const char * inMenuItemName, const char * inAfterMenuItemName,
                                  AVComputeEnabledProc inAVComputeEnabledProc,  void * inAVComputeEnabledProcData,
                                  AVExecuteProc inAVExecuteProc, void * inAVExecuteProcData )
  {
    AVMenubar   theAVMenubar ;
    AVMenu      theAVMenu ;
    AVMenuItem  theAVMenuItem ;
    AVMenuItem  theAfterAVMenuItem ;
    ASInt32     theError = 0 ;
    ASInt16     index ;

    DURING

      theAVMenubar  = AVAppGetMenubar() ;
      if ( theAVMenubar == NULL )
        E_RTRN_VOID ;

      theAfterAVMenuItem = AVMenubarAcquireMenuItemByName( theAVMenubar, inAfterMenuItemName ) ;

      theAVMenu = AVMenuItemGetParentMenu ( theAfterAVMenuItem ) ;

      index = ( ASInt16 ) AVMenuGetMenuItemIndex( theAVMenu, theAfterAVMenuItem ) ;

      theAVMenuItem = CreateMenuItem( theAVMenu, inTitle, inMenuItemName, inAVComputeEnabledProc,
                                      inAVComputeEnabledProcData, inAVExecuteProc, inAVExecuteProcData ) ;
      if ( theAVMenuItem == NULL )
        E_RTRN_VOID ;

      AVMenuAddMenuItem( theAVMenu, theAVMenuItem, index + 1 ) ;

      AVMenuItemRelease( theAVMenuItem ) ;

    HANDLER
      theError = ERRORCODE ;
      DisplayErrorAlert( theError, "Error adding new menu item" ) ;
      ASRaise( theError ) ;
    END_HANDLER

    return ;

  } // end AddAfterMenuItem

// -------------------------
// Append a newly created menu to the given menu after it creates the menu item.

AVMenuItem AppendMenuItem( AVMenu inAVMenu, const char * inTitle, const char * inMenuItemName,
                                    AVComputeEnabledProc inAVComputeEnabledProc,
                                    AVExecuteProc inAVExecuteProc, void * inAVExecuteProcData )
  {
    AVMenuItem  theAVMenuItem ;
    ASInt32     theError = 0 ;

    DURING

      theAVMenuItem = CreateMenuItem( inAVMenu, inTitle, inMenuItemName,
                                      inAVComputeEnabledProc, NULL, inAVExecuteProc, inAVExecuteProcData ) ;
      if ( theAVMenuItem == NULL )
        E_RETURN( ( _t_AVMenuItem * )NULL ) ;

      AVMenuAddMenuItem( inAVMenu, theAVMenuItem, APPEND_MENUITEM ) ;

      AVMenuItemRelease( theAVMenuItem ) ;

    HANDLER
      theError = ERRORCODE ;
      DisplayErrorAlert( theError, "Error appending menu item" ) ;
      ASRaise( theError ) ;
    END_HANDLER

    return theAVMenuItem ;

  } // end AppendMenuItem

// -------------------------
// Append a menu item to the About sub-menu.

void AppendToAboutMenu( const char * inTitle, const char * inMenuItemName, AVExecuteProc inAVExecuteProc )
  {
    AVMenubar   theAVMenubar ;
    AVMenu      theAboutAVMenu ;
    ASInt32     theError = 0 ;

    DURING

      theAVMenubar  = AVAppGetMenubar() ;
      if ( theAVMenubar == NULL )
        E_RTRN_VOID ;

      theAboutAVMenu  = AVMenubarAcquireMenuByName( theAVMenubar, "AboutExtensions" ) ;
      if ( theAboutAVMenu == NULL )
        E_RTRN_VOID ;

      AppendMenuItem( theAboutAVMenu, inTitle, inMenuItemName, ( AVComputeEnabledProc )NULL, inAVExecuteProc, NULL ) ;

      AVMenuRelease( theAboutAVMenu ) ;

    HANDLER
      theError = ERRORCODE ;
      DisplayErrorAlert( theError, "Error adding About Reverse Pages menu item" ) ;
      ASRaise( theError ) ;
    END_HANDLER

    return ;

  } // end AppendToAboutMenu

// -------------------------
//
// Callbacks
//
// -------------------------
// A generic compute enabled proc to return true if any documents are opened.
// Determine whether our command menuitem and the toolbutton are enabled
// Pass the security parameter you want to check for as the 3rd parameter of
// AVMenuItemSetComputeEnabledProc or AVToolButtonSetComputeEnabledProc.

static ACCB1 ASBool ACCB2 DoComputeEnabled( void * inPermRequired )
  {
    AVDoc theAVDoc = AVAppGetActiveDoc() ;
    if ( theAVDoc == NULL )
      return false ;

    if ( inPermRequired == NULL )
      return true ;

    PDPerms theDocPDPerms = PDDocGetPermissions( AVDocGetPDDoc( theAVDoc ) ) ;

    return ( !inPermRequired || ( ( ( PDPerms )inPermRequired & theDocPDPerms ) != 0 ) ) ;

  } // end DoComputeEnabled

// -------------------------
// Display the About box

static ACCB1 void ACCB2 DoAboutReversePages( void * ioUserData )
  {

    AVAlert( ALERT_NOTE, "Reverse Page will reverse the order of pages in the document.",
                         "OK", ( const char * )NULL, ( const char * )NULL, false ) ;

    return ;

  } // end DoAboutReversePages

// -------------------------
// Reverse the order of pages in the current active document.

static ACCB1 void ACCB2 DoReversePages( void * ioUserData )
  {
    AVDoc       theAVDoc ;
    PDDoc       thePDDoc ;
    AVPageView  theAVPageView ;
    AVCursor    theAVCursor ;
    AVCursor    theWaitAVCursor ;
    ASInt32     theNumberOfPages = 0 ;
    ASInt32     index = 0 ;

    DURING

      // change the cursor to the wait cursor
      theAVCursor = AVSysGetCursor() ;
      theWaitAVCursor = AVSysGetStandardCursor( WAIT_CURSOR ) ;
      AVSysSetCursor( theWaitAVCursor ) ;

      // get the AVDoc for the frontmost document
      theAVDoc = AVAppGetActiveDoc() ;

      thePDDoc = AVDocGetPDDoc( theAVDoc ) ;

      PDDocAcquire( thePDDoc ) ;

      // get the count of pages in the document
      theNumberOfPages = PDDocGetNumPages( thePDDoc ) ;

      // loop through all of the pages changing the page order
      for ( index = 1 ; index < theNumberOfPages ; index++ )
        PDDocMovePage( thePDDoc, PDBeforeFirstPage, index ) ;

      PDDocRelease( thePDDoc ) ;

      // display the first page on screen
      theAVPageView = AVDocGetPageView ( theAVDoc ) ;
      AVPageViewGoTo ( theAVPageView, 0 ) ;

      // change the cursor back to the system cursor
      AVSysSetCursor( theAVCursor ) ;

    HANDLER
      DisplayErrorAlert( ERRORCODE, "Error reversing pages" ) ;
    END_HANDLER

    return ;

  } // end DoReversePages

// -------------------------
//
// Plug-in setup
//
// ------------------------

static ACCB1 ASBool ACCB2 InitPlugInMenus( void )
  {
    AVMenubar     theAVMenubar ;

    DURING

      theAVMenubar  = AVAppGetMenubar() ;   // get the main menu
      if ( theAVMenubar == NULL )
        E_RETURN( false ) ;

      AppendToAboutMenu( "Reverse Pages...", "NAME_DoAboutReversePages", &DoAboutReversePages ) ;

      AddAfterMenuItem( "Reverse Pages", "NAME_ReversePages", "ReplacePages", &DoComputeEnabled,  NULL, &DoReversePages, NULL ) ;

    HANDLER
      DisplayErrorAlert( ERRORCODE, "Error installing Reverse Pages menu items" ) ;
    END_HANDLER

    return true ;

  } // end InitPlugInMenus

// ------------------------
// called by Acrobat to allow the plug-in to do any required setup.
// this is a rather simple plug-in which only needs to init the menus.
// a more complex plug-in may also need to init toolbar buttons, annotation handlers, etc.

static ACCB1 ASBool ACCB2 InitPlugIn( void )
  {
    ASBool    theResult ;

    theResult = InitPlugInMenus() ;
    if ( theResult == false )
      return theResult ;

    return theResult ;

  } // end InitPlugIn

// ------------------------
// called by Acrobat to allow the plug-in to cleanup before it is unloaded.

static ACCB1 ASBool ACCB2 UnloadPlugIn( void )
  {
    return true ;

  } // end UnloadPlugIn

// ------------------------
//  This is the pre-initialization routine for the plug-in.
//  The pre-initialization routine is installed in PIHandshake().  Use the
//  pre-initialization routine to register for any event notifications.
//  This is important in this particular plug-in since the main
//  initialization function itself causes events to occur.  Since this
//  plug-in counts various events, we don't want to miss out on any of the action.

static ACCB1 ASBool ACCB2 PreInitPlugIn( void )
  {

    return true ;

  } // end PreInitPlugIn

// ------------------------
// called from PIMain to give Acrobat all of the required callbacks.

ACCB1 ASBool ACCB2 PIHandshake( Uns32 handshakeVersion, void * handshakeData )
  {
    if ( handshakeVersion == HANDSHAKE_V0200 )
      {
        PIHandshakeData_V0200 *hsData = ( PIHandshakeData_V0200 * )handshakeData ;

        hsData->extensionName = ASAtomFromString( "NAME_ReversePages" ) ;

        hsData->exportHFTsCallback = NULL ;

        hsData->importReplaceAndRegisterCallback = NULL ;

        hsData->initCallback = ASCallbackCreateProto( PIInitProcType, ( void * )InitPlugIn ) ;

        hsData->unloadCallback = ASCallbackCreateProto( PIUnloadProcType, ( void * )UnloadPlugIn ) ;

        return true ;
      }

    return false ;

  } // end PIHandshake

// ------------------------
```