Section 5 - Fonts

Purpose

In this section you will learn about some of the MLTE API that greatly simplify the font handling in your application.

Objective

Upon completion of this section you will be able to do the following:

- Use the MLTE font editing API:

  TXNNewFontMenuObject, TXNDisposeFontMenuObject, TXNDoFontMenuSelection, TXNSetTypeAttributes

Outline

The following topics will be covered in this section:

- Creating and disposing of the font menu
- Changing the font based on the font menu selection
- Setting font attributes like size and style

Creating the Font Menu Object

MLTE provides utility functions for creating and handling a standard font menu.

MLTE provides an opaque structure called **TXNFontMenuObject** that can be used to handle user interaction with the font menu. The menu is created dynamically with **TXNNewFontMenuObject**:

```
EXTERN_API(OSStatus)TXNNewFontMenuObject(MenuHandle iFontMenuHandle,
                                         SInt16 iMenuID,
                                         SInt16 iStartHierMenuID,
                                         TXNFontMenuObject* oTXNFontMenuObject);
```

You provide the menu handle, the menu ID, and the menu ID to use if any hierarchical menus are created (on systems with ATSUI). A new TXNFontMenuObject is returned.

The **iFontMenuHandle** is an empty menu handle.

The **iMenuID** is the menu ID that the font menu should have.

The **iStartHierMenuID** is the menu ID at which hierarchical menu IDs will begin.

A good place to call TXNNewFontMenuObject is at the same time you are preparing to display your menubar. In the sample, this is at Step 27 in the Initialize function. The sample already provides a global, gTXNFontMenuObject, to hold the returned TXNFontMenuObject.

Add the code to create a new TXNFontMenuObject. Your code might look something like this:

```
status = TXNNewFontMenuObject(GetMenuHandle(mFont), mFont, kStartHierMenuID,
                              &gTXNFontMenuObject);
if (status != noErr)
      BigBadError(eNoCreateFontMenuObject);
```

Notes

Disposing the Font Menu Object

Once created, you must arrange to dispose of the TXNFontMenuObject. MLTE provides the function **TXNDisposeFontMenuObject** for this task:

```
pascal OSStatus TXNDisposeFontMenuObject(TXNFontMenuObject iTXNFontMenuObject);
```

Note that this function calls DisposeMenuHandle on the Font menu handle for you.

A good place to call TXNDisposeFontMenuObject is in the sample's Terminate function, at Step 28. Your code might look something like this:

```
if (gTXNFontMenuObject != NULL)
{
   status = TXNDisposeFontMenuObject(gTXNFontMenuObject);
   if (status != noErr)
      AlertUser(eNoDisposeFontMenuObject);
   gTXNFontMenuObject = NULL; // Nullify font menu object
}
```

Handling the Font Menu

Now that you have a font menu, you need to handle any selections that the user makes from it. MLTE provides a function, **TXNDoMenuSelection**, to to this:

```
EXTERN_API(OSStatus)TXNDoFontMenuSelection(TXNObject iTXNObject,
                           TXNFontMenuObject iTXNFontMenuObject,
                           SInt16 iMenuID,
                           SInt16 iMenuItem);
```

Change the currently selected text to the font the user selected.

Our sample app calls its own DoFontMenu() function when appropriate. At Step 29, check that the front window belongs to your app and, if so, call the TXNDoFontMenuSelection function. Your code might look something like this:

```
if (IsAppWindow(window))
{
   GetWindowProperty(window, 'GRIT', 'tObj', sizeof(TXNObject), NULL, &object);
   if (gTXNFontMenuObject != NULL)
      status = TXNDoFontMenuSelection(object, gTXNFontMenuObject, menuID, menuItem);
   if (status != noErr)
      AlertUser(eNoFontName);
}
```

Setting Font Attributes

Common attributes that you will want to be able to set for selected text are the font size and the font style. MLTE provides the function TXNSetTypeAttributes to make this possible:

```
EXTERN_API(OSStatus)TXNSetTypeAttributes(TXNObject iTXNObject,
                                         ItemCount iAttrCount,
                                         TXNTypeAttributes iAttributes[],
                                         TXNOffset iStartOffset,
                                         TXNOffset iEndOffset);
```

You can use TXNSetTypeAttributes to set the current selection's (or some range that you specify) font information.

The **iAttrCount** parameter specifies the number of type attributes in the TXNTypeAttributes array.

The **iAttributes[ ]** parameter is an array of attributes that we would like to set.

```
struct TXNTypeAttributes {
   TXTNTag tag;
   ByteCount size;
   TXNAttributeData data;
};
typedef struct TXNTypeAttributes TXNTypeAttributes;
```

Constants for the TXNTypeAttributes tag and size members are defined in MacTextEditor.h.

The parameters **iStartOffset** and **iEndOffset** delineate the range of text in which you want to set the attributes. If the goal is to change the current selection, these should be **kTXNUseCurrentSelection**.

Notes

Our sample app has a "case mSize:" switch in its DoMenuCommand function. At Step 30, check that the front window belongs to your app and, if so, call the TXNSetTypeAttributes function to set the current selection to the user's size selection from the Size menu. Your code might look something like this:

```
// Constant specifying size attribute -- FOUR_CHAR_CODE('size')
typeAttr.tag = kTXNQDFontSizeAttribute;
// Constant specifying the size of the size attribute -- sizeof(SInt16)
typeAttr.size = kTXNQDFontSizeAttributeSize;
// move data into high byte
typeAttr.data.dataValue = shortValue << 16;

GetWindowProperty(window, 'GRIT', 'tObj', sizeof(TXNObject), NULL, &object);
// Set the size attributes
status = TXNSetTypeAttributes(object, 1, &typeAttr, kTXNUseCurrentSelection,
                              kTXNUseCurrentSelection);
```

Our sample app has a "case mStyle:" switch in its DoMenuCommand function. At Step 31, check that the front window belongs to your app and, if so, call the TXNSetTypeAttributes function to set the current selection to the user's style selection from the Style menu. You might specify your settings something like this:

```
typeAttr.tag = kTXNQDFontStyleAttribute;
typeAttr.size = kTXNQDFontStyleAttributeSize;
typeAttr.data.dataValue = newStyle;
```

Build and run the MLTESampleShell application. Try out the Size and Style menu items.

Notes

Notes