Extension Development
Fantasy Football for Windows
(c) 1993, PAS Software

INTRODUCTION

Many software applications that target the fantasy software market tend to present one or two methods of league operation. Through user research, it is easy to recognize the vast diversity of methods used to implement a league. As a result, PAS Software has attempted to create an application that is not only highly customizable as is, but usable for any league implementation through the use of 3rd party extensions.

As a registered user, you are entitled to develop your own fantasy extensions. You are entitled to distribute these extensions and require registration fees, etc.

What is an extension? An extension is a Windows dynamic link library. But, by linking to PASOBJ.OBJ, the requirement to manage complex interfaces, internal memory management, and fantasy application/extension communication. Additionally, the need to process WinMain() and WEP() functions are eliminated. Not only does this make the developers life easier (the development of an extension requires minimal Windows 'C' programming experience), but protects the fantasy application from crashes.

If you've chosen to install the Extension Development Kit, you have also received a sample extension. To install the extension, simply copy the extension to the directory in which Fantasy Football for Windows resides and restart the application. The extension demonstrates basic principles used to customize the app and even exports player statistics.

TECHNICAL DETAILS

Up to date technical details can be found in the included help file, PASEXT.HLP. This file includes specifications for the Sport Fantasy Application Programming Interface (SFAPI). SFAPI will be used across all PAS fantasy applications and will be fully maintained in the future. Further, this help file contains structure details and variable descriptions.

Building Extensions

Extensions require to be developed in large-model. PASOBJ.OBJ has been developed and compiled with Microsoft MSVC++ (also known as C 8.0). The sample's included makefiles are geared towards working with Microsoft MSVC++. NOTE: At this time, no other compilers have been tested with other compilers or linkers. Please contact PAS Software with any problems.

To build the sample extension, simply invoke NMAKE from the command line.

Necessary Files

During installation, several files were copied to the specified destination. The following files are required to build extensions:

($DESTINATIONDIRECTORY)\LIB\PASOBJ.OBJ
($DESTINATIONDIRECTORY)\LIB\CTL3D.LIB
($DESTINATIONDIRECTORY)\H\PASEXT.H

For simplicity, the sample extension builds with these files using hard-coded paths. Usually, the required files would be copied to your own include and library directories. Doing so, the sample source code would require the removal of these hard-coded paths.

SAMPLE EXTENSION

Included with this installation is a sample extension, including source code. The extension demonstrates most of the major functionality an extension has access to. Further, the source code attempts to demonstrate many of the principles necessary for developing quality extensions.

The following describe standard extension functionality that an extension can modify and are located in CUSTOM.C:

* **CustomPrint()** provides the functional code for printing two additional printing options the extension makes available to the user. The first option simply prints multiple pages and demonstrates page number access and the necessary steps to force a page eject. The second option cycles through the available players and prints all those players that rushed for 100+ yards.

* **CustomStandings()** just calls DialogBox() with the necessary dialog resource identifiers and attaches the StandingsMsgProc() dialog procedure. The StandingsMsgProc() simply cycles through the available fantasy teams and adds them to the displayed list box. Usually, the extension would sort these teams according to some criteria and then display them to the user.

* **CustomStat()** provides the functional code for displaying two additional statistical options the extension makes available to the user. The first option generates a list of 100+ yard receivers. The second option generates a list of 100+ yard rushers.

* **PlayerWeeklyScore()** modifies the application's fantasy score calculation procedure. While much more sophisticated scoring calculations are used, the sample simply determines if a player has rushed more than 100 yards for the given week. If the player pinnacles 100 yards, the player receives 3 points. Otherwise the player receives zero.

* **PlayerWeeklyYards()** modifies the application's fantasy yard calculation procedure. This extension simply demonstrates the principle by assigning each player 10 yards for each given week in achieved yards.

* **PrintOptions()** demonstrates how additional print options are added to the print option dialog box that is presented to the user. This sample adds the selections "Custom Print Options" and "Custom Print 100+ yard rushers" to the list box.

* **StatOptions()** demonstrated how additional statistical options are added to the team points dialog box that is presented to the user. This sample adds the selections "Custom 100+ yard receivers" and "Custom 100+ yard rushers" to the list box.

* **TransactionNotify()** demonstrates how transaction notification messages are trapped and processed. The sample traps a transaction notify message, determines the team to which the transaction is to be charged, and displays a dialog box notifying the user of the transaction cost. Usually, this function would save these transaction costs for yearly accumulations.

* **WeeklyScores()** just calls DialogBox() with the necessary dialog resource identifiers and attaches the WeeklyScoreProc() dialog procedure. The sample requires that the user's team has been previously identified and displays the number of power points their starters have scored and the number of power points their reserves have scored. Further, it demonstrates the ability to modify the displayed week and bound checking methods.

The following describe required functionality for each and every extension. These functions are located in MAIN.C:

* **InitExtensions()** is a mandatory function that is called by the fantasy application when the extension is first loaded. While the sample extension does nothing, usually an extension would do some initialization of variables or memory.

* **GetCaps()** is a mandatory function that is called by the fantasy application that determines what type of custom functionality an extension supplies. Because the sample demonstrates all custom functionality, it returns EXT_ALL.

* **InitMenus()** is a mandatory function that is called by the fantasy application when the extension is first loaded. The sample extension adds a pop-up menu, with the text "Extension", and adds a total of three menu options, "Performance", "Export", and "About". Further, the sample demonstrates how to add button capabilities.

The following describe functions that are supplied as menu functionality to the user. These functions are located in FUNCS.C:

* **Performance()** just calls DialogBox() with the necessary dialog resource identifiers and attaches the PerformMsgProc() dialog procedure. The PerformMsgProc() cycles through the players and generates a list of the top three yardage leaders for the current week in the quarterback, running back, and wide receiver positions.

* **Export()** just calls DialogBox() with the necessary dialog resource identifiers and attaches the ExportMsgProc() dialog procedure. The ExportMsgProc() allows the user to select a specified week in which player statistics are exported in a tab-delimited text file.

* **ExportASCIIText()** actually does the grunt work of cycling through the available players and exporting the player statistics to a tab-delimited text file. This function can be modified to export statistics into a number of known binary formats as well.

* **About()** simply creates a dialog box with strings describing the extension and extension development.

AUTHORIZED EXTENSIONS

PAS Software will be distributing the fantasy applications on a nation-wide basis. In an effort to document existing extensions and maintain quality, PAS Software will be authorizing extensions. If an extension is authorized, the extension will be distributed as part of a supplemental extension disk to registered users. All authorized extensions will be evaluated for inclusion into the final distributed version of the product as well.

For authorization, the extension, along with detailed functional specifications, must be sent to PAS Software. PAS Software will put the extension through a rigorous testing procedure and verified for minimal functionality/quality. To submit an extension, please send to:

PAS Software
ATTN: Extension Development
16625 Redmond Way, Suite M333
Redmond, WA 98052
Phone: (206) 788-7568
FAX: (206) 867-0737
BBS: (Available August 1, 1993)