Vista Support

CheckForVista

A function in the Sexy namespace has been added to query if a program is running on Vista. This function is used by the framework and several PopCap games to perform Vista-specific processes that differ from previous operating systems in some way. For example, to support backwards compatibility with previous framework versions, the latest release does not call **SetAppDataFolder** unless the program is running on Vista. This ensures that all data will be written to and read from an expected location on XP using the newest framework as well.

CheckFor98Mill

This is a very similar function to **CheckForVista**. However, it is used to detect Windows 95, 98, and ME. This function has little to do with Vista Support in the framework. It is actually there to guide behavior of the framework when _USE_WIDE_STRING is turned on. Certain calls like CreateWindowExW are not properly supported on these operating systems unless the Microsoft Layer for Unicode is installed. As a result, some different code paths are taken in various locations around the framework if this returns true.

GetAppDataFolder and **SetAppDataFolder**

To properly support Vista, programs are forbidden to write data to several locations. One of these locations is *Program Files*, the typical location for installation of games and other programs on Windows. Accordingly, the framework will call **SetAppDataFolder** during SexyAppBase::Init to ensure the **GetAppDataFolder** function is set up correctly for any reading or writing of data to the hard disk.

Whenever a program needs to read or write data, it is expected to do this by first calling **GetAppDataFolder** and then appending the filename it wishes to write. For example, if a program wants to write to a file which would have been called *"userdata/users.dat"* in previous versions of the framework, it should now write the file using **GetAppDataFolder() + "userdata/users.dat"** On Windows versions prior to 6.x (Vista), **GetAppDataFolder()** will return an empty string, thus producing the exact same result.

The framework itself has already been modified to use this new methodology in all locations it writes files. This includes the cached wave files that get written during sound loading calls.

How do I override the application data folder to include my company's name?

There is a new member variable in SexyAppBase called **mFullCompanyName**. This variable is used in the initial call to **SetAppDataFolder**. First, the framework queries the operating system using the Windows API SHGetFolderPath for the COMMON_APPDATA folder. Then, the final application data folder set is: **%COMMON_APPDATA%\mFullCompanyName\mProdName**, where **mProdName** is some internal name of your application. PopCap typically uses the name of the game (sans the word Deluxe) and without spaces for this variable. For example, this will eventually become c:\ProgramData\PopCap Games\Bejeweled2 if the game being built is Bejeweled 2 Deluxe.

AllowAllAccess

AllowAllAccess is a function used to set up ACL information giving the "Everyone" group "Full Control" access of a file or directory. It should be supported on any operating system that supports NTFS-formatted partitions. PopCap uses this function to properly support XP to Vista upgrade paths, as well as setting file security when some files are first created. For example, in Bookworm Adventures, the .bwa files used to save profile data have this function called on them immediately after creation. This ensures that all users (Limited, Standard, and Admin) will be able to read and write to them. In other games, like Bejeweled 2, the program data directory is given all access immediately after it's created (in the installer as well as after data is moved when an upgrade is detected). The files under this directory then inherit the permissions directly from the directory.

Windows Registry Changes

All registry keys in the framework are written to and read from HKEY_CURRENT_USER now, regardless of operating system. There is no support for backwards compatibility in this regard. The framework will neither automatically remove any previous HKEY_LOCAL_MACHINE values written, nor will it read them as a back-up if HKEY_CURRENT_USER values do not exist. It is up to your program's (un)installation engine to properly handle this.