Solitaire 3.1

OpenStep-Mach/Rhapsody/YellowBoxForWindows release

Since all the previous maintainers of Solitaire have gone on to other platforms, I thought someone better make sure that Solitaire lives on for OpenStep and Rhapsody.

History

The first release of Solitaire was made by Blake Stone <bstone@dkw.com> a long time ago (~1991). From there, before he worked for the evil empire, Gary Ritchie <garyri@microsoft.com> took it, added the game bundle concept and numerous other card games to the package, including FortyThieves, Pyramid, Spider and Tenpile. Then OpenStep came along and the package was converted seperately by both Todd Thomas <tthomas@cmg.fcnbd.com> and Bill Chin <s0wwchin@atlas.vcu.edu>. Our efforts were merged and we came up with this package.

Changes from Previous Releases

During the initial OpenStep converson the CardSet objects were enhanced. See ReleaseNotes.rtf in the Solitaire framework's CardSet.subproj for more information. I also made the objects in the CardSet project autodoc friendly, in order to make it easy to generate new documentation when changes to the objects are made. Thanks to Adam Swift (aswift@friday.com) for making autodoc freely available and making documentation generation almost painless (autodoc can be found in the Examples directory of the MiscKit from versions1.4.x to 1.7.x).

I'd also like to thank Erik Kay <email unknown> and Matt Ash <mash@cmg.fcnbd.com> for changes necessary to make Solitaire compile and work right on OpenStep NT, and both Erik Kraft and David Fedchenko for their bug fixes, ideas and code enhancements to the game engine and the various games.

Have fun and feel free to email any suggestions for improvements (developer or user wise).

Building and Installation

If you received this package in source form you'll need to compile the framework,

application and the various game modules before you can play. The following steps will build the various pieces and install them:

1. Open PB.project in the Frameworks/Solitaire framework and set the install path in the inpsector. The default is $(HOME)/Library/Frameworks. Set the build target to install and build.
2. Open the top level PB.project in the Games subdirectory. The default install path for each game is $(HOME)/Library/Solitaire. If you change the install path for the games you neeed to selet each game individually in Project Builder and change it for each game. Note, for those of you with OPENSTEP, that if you still have the NEXTSTEP version of Solitaire then it searches for games by default in the same location as this application (~/Library/Solitaire and /LocalLibrary/Solitaire). If you want both games to peacefully coexist it's better to install the games somewhere else and change this application's search path completely. Instructions on how to do this can be found in Optional Step 3 below. By default the engine searches for games in ~/Library/Solitaire, /LocalLibrary/Solitaire, and the the Solitaire app wrapper. If you decide to install it any other place you'll neeed to tell the engine where to look. Instructions on how to do this are in are below in Step 5.
3 [Optional if you have the NEXTSTEP version of Solitaire and want them to peacefully

coexist] I suggest that you change the application source code before you build the application. In the source code (Solitaire/Solitaire) you should find a file called SolitaireMore.m. The first method is called -getGameType. In it is where you define your paths to find the game bundles. For example you could change the search paths to ~/Library/Solitaire-os/ and /LocalLibrary/Solitaire-os/ so that the OpenStep Solitaire.app only finds the OpenStep game bundles and not the NEXTSTEP ones (since they won't load anyways).
4. Open PB.project in the Solitaire app directory and set the install path in the inspector. The default is $(HOME)/Apps. If you installed the above Solitaire.framework in some place other than $(HOME)/Library/Frameworks then you'll also need to add that path to the project's framework search paths so it can be found during building. This can be done from the inspector. Take note that if you build this on Windows that you'll need to remove the SoundKit framework from the list of dependant frameworks before building. Finally, set the build target to install and build.
5. [Optional if you installed the Games somewhere other than ~/Library/Solitaire, /LocalLibrary/Solitaire or the app wrapper and didn't do Step 3 above]. The Solitaire game engine needs to know where the games are located. There's a user default, SolitaireBundlePaths that can be used to add game search paths. Just type this at the command line: defaults write Solitaire SolitaireBundlePaths

/Whereever/You/Installed/Games;/AndHereToo/Solitaire or on Windows c:\winnt\system32 (isn't that where everything goes on Windows :-)

Platforms

This app compiles and has been tested under OPENSTEP4.2/Mach, RhapsodyDR1/Intel, and Yellow Box for Windows. It will also likely work fine under RhapsodyDR1/PPC. It may also work under OpenStep4.2/NT too.

Todd Thomas <tthomas@cmg.fcnbd.com>
Bill Chin <s0wwchin@atlas.vcu.edu>
February 1998