The Steem 3.2 B Source Tree

Version 3.2B is based on the original 3.2A Steem sources downloaded December 1, 2011.

The goal of this version is to completely reorganize the sources of Steem. In the original Steem project all the .h **and** .cpp files were included from 3 central .cpp files (Steep.cpp, emu.cpp, helper.cpp). This is an extremely unusual way to organize the sources that render the development with a modern IDE particularly unfriendly (for example browsing the sources or debugging the code). This organization is also not suitable for documentation tools like Doxygen.

I have therefore decided to create a new source tree that fixes this structural problem without changing any of the existing functionality and at the same time I have added minimum Doxygen information so basic documentation can be produced. ***The changes has been kept has minimal has possible from the original so that difference tool can be used to compare sources***.

Hopefully the 3.2B source tree is now organized as it should have been. The .h files only contain the declarations for the corresponding .cpp files that only contain the implementation.

This was a **huge effort** because everything was completely mixed up: you had a lot of declarations done in .cpp file but the worst was the fact that a lot of code implementation as well as variable initialization were done in the .h file! This has forced me to move/duplicate a lot of information from the .h files to .cpp files. There were also a lot of declarations missing from the .h files as everything was included in the 3 central files. There was also an interesting case were two different macros were defined with the same name (PC_SET) and the selection of one definition versus the other was done based on information in the macro IN_EMU. Fixing this macro was challenging and was verified by comparing the outputs of the fully preprocessed central files!

Directory tree modification

The Steem source tree has been kept as close as possible with the original:

- The ***3rdparty*** directory is unchanged from original.
- The ***include*** directory contains the sources files that are somewhat independent of Steem. Many files have been changed or added in this directory. This directory includes an **asm** and **x** directories that have not been touched. There is also a new directory called **not_used** that contains all the files currently not used in the Windows Steem build.
- The source code from the **steem** directory (steem.cpp, emu.cpp, helper.cpp) has been displaced to a **not_used** directory. Under steem we find the following directories:
    - **asm** assembly code not modified
    - **code** directory: contains all the sources and all of them have been modified. This directory also contains a **x** directory that has not been touched and a **not_used** directory where all the code unused in the current Windows build has been moved (this include the **draw_c** directory has new draw_c file have been placed in the **code** directory).
    - The **doc** directory has not been changed but now contains a new subdirectory called **jlg** that contains all my documentation.
    - The **lib** directory: Contains theunzipd32.dll. I have added in this directory the dinput8.lib and dxguid.lib (part of Microsoft DirctX SDK) required to build steem;
    - The **macros, patches, rc, shortcuts** directories have not been changed
- **Windows-build** and **X-build** directories have not been changed
- A new VC2010 directory has been created that contains the VC++ 2010 configuration files.

In the root directory you will also find the Visual C++ 2010 solution files for the Steem project.

What has been done:

- Splitting of declaration/implementation code in .h and .cpp files.
  For that matter new files have been created when necessary:
  - Some .cpp files have been created from .h to place static variables and initialization when no corresponding file existed and
  - Some .h files have been created from .cpp files to place variable and function declaration when necessary.
- The resulting code has been compiled in debug and release mode. This has been done only on Win32 platform using VC++ 2010 (procedure detailed elsewhere).
- Minimal Doxygen information has been added to sources in order to produce documentation with Doxygen and Graphviz. The functions in modules have been organized in 4 groups: **Drawing**, **Core and GUI**, **Emulation**, **Helper**. As the internal documentation of Steem is almost nonexistent, this is better than nothing.
- The compilation in the original tree was done with all warning off. I have turned the Warning detection in VC++ to Level 3 (this is the usual level) and I have fixed all the warnings with few exceptions (look for jlg_cxxxx in the code).
- Very basic tests in all resolutions of the debug and release tree
- In the header of each file a description of the module, the GNU GPL3 license agreement, as well as RCS variables have been added.
- The _DEBUG_BUILD / _NO_DEBUG_BUILD macros were confusing has they did not relate at all with building debug/release version of Steem. I have therefore replace the _DEBUG_BUILD macro with the more explicit STEEM_DEBUGGER macro

What has not been done:

- The UNIX code in the source has not been processed. This means that anything not defined for Windows has not been changed but was not removed. Therefore if someone wants to use this tree in UNIX it has to do the work.
- The code has been only tested with MS VC++ 2010. No other compiler has been tested.
- Only very few conditional macros has been tested.
- The ONEGAME macro has been “processed” but the resulting code does not compile yet.
- Only basic tests of the program have been done so far. Things may have been broken during the reorganization and more checking and tests need to be performed.

Planned for future release:

- There is a Warning in debug_emu (line 272) that looks more like a bug and needs to be investigated
- Remove/Replace some Macros by inline functions. See “Macros are evil” at http://www.parashift.com/c++-faq-lite/misc-technical-issues.html#faq-39.4.
  Personally I very rarely use macros as they are potentially unsafe if not properly used and not user friendly in debug environment. However there are some places where it would be very difficult not to use them.
- Continue cleaning of the code. There are still a lot of places where the declaration are misplaced or missing
- Documentation. Would be nice to add minimum documentation but this is a daunting task!
- Warning unsafe function
- Warning Wrong declaration and/or conversion
  - Performance warning,
  - Unsafe mix of type 'int' and type 'bool' in operation, etc.)
  - conversion with possible loss of data
  - truncation of constant value in initialization