

SuperCollider and cross platform issues

Marije A.J. Baalman

Overview

- Platform?
- Editor
 - Document interface
 - Graphical user interface
 - Helpfile interface
- System/OS backend
 - Compilation procedure
 - Abstraction in primitive source code
 - Abstraction in SuperCollider source code

What is meant with platform?

SuperCollider can currently be compiled for four different operating systems^a (OSX, Linux, Windows and FreeBSD). But, that is not the only issue...

SuperCollider consists of two parts (*sclang*, *scsynth*)...

^apossibly more, if they adhere to POSIX standards

What is meant with platform?

SuperCollider can currently be compiled for four different operating systems^a (OSX, Linux, Windows and FreeBSD). But, that is not the only issue...

SuperCollider consists of two parts (*sclang*, *scsynth*)...

... well, actually three (the *editor* environment)

^apossibly more, if they adhere to POSIX standards

What is meant with platform?

SuperCollider can currently be compiled for four different operating systems^a (OSX, Linux, Windows and FreeBSD). But, that is not the only issue...

SuperCollider consists of two parts (*sclang*, *scsynth*)...

... well, actually three (the *editor* environment)

... and sometimes four (the *graphical* server)

^apossibly more, if they adhere to POSIX standards

What is meant with platform?

SuperCollider can currently be compiled for four different operating systems^a (OSX, Linux, Windows and FreeBSD). But, that is not the only issue...

SuperCollider consists of two parts (*sclang*, *scsynth*)...

... well, actually three (the *editor* environment)

... and sometimes four (the *graphical* server)

... and potentially even more (e.g. *scgraph*)

^apossibly more, if they adhere to POSIX standards

Editors

- *SCapp* - the native Cocoa editor on OSX.
- emacs (*sce/*) - emacs plugin.^a
- vim (*scvim*) - vim plugin.^a
- gedit (*sced*) - gedit plugin, Linux only.
- *Psycollider* - Python based editor.^b
- *JSCEclipse* - Java based IDE.^b
- *TextMate* - TextMate plugin/bundle. OSX only.
- Other editors: *scfront*, *qcollider*, *squeak*...

^aDeveloped for Linux, but can be set up on OSX, and theoretically also Windows.

^bDeveloped for windows, but can run on other platforms as well.

Editors and OS

From the previous list we can see that editor environment is not really dependent on operating system.

In other words:

OSX does not always equal *SCapp*

nor does Linux always equal *sce/*, and Windows always equal *PsyCollider*.

Even when these are the predominant cases...

*We should think of a way to make “platform” specific switches based on the editor in which **sclang** is running.*

Editors and Document interface

- Since *SCapp* has compiled *sclang* into its program, the calls to the GUI and the Document interface are within the same application, so there are no problems with synchronicity.
- Psycollider also compiles *sclang* into itself, so Document and maybe GUI calls would be possible to be synchronous.^a
- Most of the other editors invoke *sclang* as a terminal client, and pipe commands to it to execute code, and catch its output to be printed in a post window.
- Of the other editors, only *sce/* has (partial) Document support. Problems occur where calls to the document interface expect a direct reply (e.g. getting the string).

^aProbably the same for JSCEclipse

Editors and graphical interfaces

- *SCapp* has GUI calls integrated, the *CocoaGUI*.
- The other editors use SwingOSC as an external GUI server (*SwingGUI*).^a.

The GUI abstraction layer solves most of the compatibility issues.^b

*But there are some issues where things **blur**...*

^aThough *sce/* also features the possibility to create Emacs-buffer based user interfaces. However currently not compatible with the GUI abstractions. But it is relevant for some guis (*synthdef*, *server*, ...).

^bas long as we keep new additions coordinated between SwingOSC and Cocoa, and always use GUI.id if we need to check to set a specific feature based on GUI kit.

Editors and menu interfaces

- Since *SCapp* integrates *sclang* into its program, it can add Menu items to its application menu.
- However, this would be equally possible for *sce/*, and possibly other editors...
- *SwingOSC* also allows for adding Menu items to its GUI, but that is then a menu in that specific GUI window. This can be useful, but it has not the same effect as Menu items in *SCapp*.

GUI kits and HID

- There exists MouseX and JMouseX. However MouseX is a server plugin. On Linux^a and Windows it works perfectly fine without a SwingOSC substitute. Same goes for the MouseY, MouseButton and KeyState UGens. On Windows the SwingOSC substitute is no longer needed.
- Wacom has a tablet GUI based interface in both Cocoa and SwingOSC, which both only work under OSX. On Linux however, Wacom tablets are accessible as a standard HID interface, and there is no need for a GUI.

^aas long as the plugins were compiled with X support

Editors and help file access

- *SCapp* - integrated; HTML parsing through WebKit.
- *scel* - via *w3m-el*. Unfortunately this built-in browser does not support CSS.
- *scvim* - strips HTML from help files during installation, and thus uses plain text.
- *sced* - opens help file in external browser (does not allow for code execution).
- *PsyCollider* - shows HTML file and can convert it to a normal code window.
- *JSCEclipse* - ??

Read-only preview is possible through Help.gui for all editors.

Editors and help file editing

- *SCapp* - integrated; HTML parsing through WebKit, as well as creation. Unfortunately, this creates hard to read HTML code.
- Other developers have to use other HTML editors to create helpfiles, which can be cumbersome, especially with regard to creating examples (color coding).
- Helper and AutoHelper are a step towards a solution, but a few more are needed to make this really work...
- A possible solution might be to allow for an alternative help file template...

Templates in SC-native format, rendering to different formats... Cleaner HTML? ... XML?

Compilation across different OS's

- *OSX* - uses XCode
- *Linux/FreeBSD* - uses a Scons script (OSX and Windows can also make use of this. Platform detection is done in the script).
- *Windows* - uses VisualC++ (maybe also mingw?)

Throughout the source code you can find platform specific defines to include or exclude certain bits of code (`SC_DARWIN` for OSX, `SC_LINUX` for linux, `SC_FREEBSD` for FreeBSD, `SC_WIN32` for Windows). This is based on different interfaces to the OS and certain libraries and headers.

OS dependencies in primitive source code

Sometimes spread over several files, sometimes within one file.

- *Audio* : `SC_CoreAudio.cpp` for OSX (with `SC_AUDIO_API_COREAUDIO` define), `SC_Jack.cpp` for Linux, `SC_CoreAudio.cpp` for Windows (with `SC_AUDIO_API_PORTAUDIO` define)
- *MIDI* : `SC_CoreMIDI.cpp` for OSX, `SC_AlsaMIDI.cpp` for Linux, `SC_PortMIDI.cpp` for Windows
- *HID* : `SC_HID.cpp` for OSX, `SC_LID.cpp` for Linux
- *Wii* : `SC_Wii.cpp` for OSX and Linux, with a lot of platform switches

S dependencies in primitive source code (2)

Sometimes spread over several files, sometimes within one file.

- *SerialPort* : `PyrSerialPrim.cpp` for OSX and Linux.
No platform switches
- *Unix* : `PyrUnixPrim.cpp` for OSX, Linux and Windows. With some switches for Windows
- *Speech* : `SC_Speech.cpp` for OSX only (not included in compilation for other platforms)

OS dependencies in class code

- *Audio* : No difference in access
- *MIDI* : Same class interface; some extra functionality in Linux. Some issues not working in Windows
- *HID* : Different class interface. Abstraction layer through GeneralHID for common interface. Windows implementation depends on external python based program (hidserver)
- *Wii* : Same class interface
- *SerialPort* : Same class interface. Not working in Windows
- *Unix* : Same class interface
- *Speech* : OSX only

Other OS dependencies

- Default installation location (and its writability)
- Default user support directory
- Default recordings directory
- Default startup file (name and location)

For all these there is platform independent access possible through the Platform class.

Class compilation

OS specific compilation of classes can be performed by putting them into OS specific directories: *osx*, *linux*, *windows*

- If you need to check for a specific OS, use `thisProcess.platform.name`.
- If you need to check for a specific GUI kit, use `GUI.id`.
- If you want to check if you are running in Emacs, use `thisProcess.platform.hasFeature(\emacs)`.

You cannot check for a class that is not compiled!

`GUI.current == CocoaGUI` will throw an error (if not on OSX), as does

`thisProcess.platform == LinuxPlatform` (if not on Linux).

Tips for writing code

In addition to the tips on the previous slide:

- If you are unsure whether a class is OS specific, look at its helpfile and/or its source file, and the location of the latter.
- GUI kit classes cannot be subclassed directly.
- Look at the Crossplatform helpfile.
- If possible, try to make your code compatible across platforms.
- If you are writing a class and it is not possible to make it cross platform, then store it in an OS specific folder. This is especially important if your class is going to be part of the main distro, the sc3-plugins, or the quarks.

Tips for writing help files

- Keep in mind that different editors have different shortcuts. Especially in Tutorials refer to the Shortcuts helpfile to point users to the fact that different shortcuts exist.
- Keep code platform independent, i.e. use GUI and other abstractions wherever possible.
- If you are writing about a feature that works in a specific editor/OS environment, please mention this clearly.

Conclusion

- *Attempts*
 - an overview of the cross platform issues within SuperCollider
 - point to some confusing issues
- *Distinction:* platform, editor, gui kit, operating system, ...
- *Discuss*
 - editor specific switches within class compilation
 - GUI kit specific switches within class compilation
 - improvement of the help file access/editing/creation situation.
- *Work to do:* implement missing base features in Windows