Self 4.3 Release Notes

Adam Spitz, Alex Ausch, and David Ungar

June 22, 2006

1. Introduction

Self is a prototype-based dynamic object-oriented programming language, environment, and virtual machine centered around the principles of simplicity, uniformity, concreteness, and liveness. We have decided to do a new release because we have ported the virtual machine to the x86 architecture, so that it can run on the new Intel-based Macintosh computers (Mac Mini, MacBook, iMac). The SPARC and PowerPC architectures are both still supported as well. Our (beta-quality) x86 port has at least one known bug: occasionally the graphical user interface freezes up. When this happens, you can type in "_Flush. desktop go" to the terminal window to unfreeze it. On the plus side, running on a 2.1 Ghz MacBook Pro, Self feels about three times faster than on a 1.7Ghz PowerBook.

Although our code is completely independent of theirs, we would be remiss if we did not mention Gordon Cichon and Harald Gliebe, who have also done an x86 port of Self. Their port (which is available at http://www.gliebe.de/self) runs on both Linux and Windows (which ours does not, yet - we would be thrilled if some kind soul were to port this latest version of Self to either of those platforms).

The Self project has not been active at Sun since 1995, although it has been used for other research. In order to make it possible for more people to experience the system, and to support our own work, we have ported Self to Mac OS X (now 10.4), for both the PowerPC and x86 architectures. Kristen McIntyre helped with the initial port. Michael Abd-El-Malek worked hard on the PowerPC port of the optimizing compiler. We have not fully integrated the Self environment with Mac OS X, but we have used it to do many hours of work on Powerbooks (and now the new Mac Minis and MacBooks!). We are releasing it so that friendly and determined users can experiment with the system, and so that Self devotees can exploit the portability framework and VM cleanup work we have done. However, this system is far from polished; consider it a beta version. (We say this every release.) We wish we had more time to file off the rough edges, but believe there is enough value added since 4.2.1 to put out another release.

Following are some brief explanatory notes. Send questions to adam.spitz@sun.com or david.ungar@sun.com. You may also want to try sending mail to the Self-interest mailing list, self-interest@yahoogroups.com.

2. Running Self (quick instructions)

On Mac OS X, start Self from the Finder by dropping a snapshot file on the SelfDroplet.

On Solaris/SPARC™, start Self by going to the `${SELF_WORKING_DIR}/objects` directory and typing the command: `./Self -s <path to a snapshot>`

More detailed instructions are below, in section 4.

3. What's Changed Since Self 4.2.1

Following is a list of the major changes since the previous release of Self. This release also includes numerous small bug fixes and enhancements; a detailed list of changes is included in Appendix A of this document.

UI1 Revival

Dave has revived the original Self user interface ("UI1"), which was built to show how cartoon animation techniques could enhance the user's experience. These cartoon animation techniques have not yet been integrated into the default Morphic user interface ("UI2"), but you can see UI1 if you have X11 installed on your computer. On a Macintosh, you will need to start up the X server, go into X's Preferences, change it to 256 color mode, and then restart the X server. Then start Self, and type the expression `ui startOn: ':0'` to start up UI1.

Salience-Based Highlighting

Self now implements David Ungar and Kristen McIntyre's patented (U.S. patent #6842182) salience-based highlighting. When selecting text, the selected text will be highlighted in a brighter color if the selected region is small (to make it easier to notice), but in a dimmer color if the selected region is large (to avoid being too intense).

Graphical Profiler

We have created an outliner to graphically display the results of running the performance profiler. (Call `profileSlice` on a block to invoke it.) It includes a pruning algorithm which attempts to eliminate uninteresting nodes from the profile tree. (You can fiddle with the settings by pressing the Controls button on the profile morph.) Both the user interface and the pruning algorithm are still very rough; we haven't had a lot of time to spend on them. If you have any suggestions or comments, please e-mail alex.ausch@gmail.com and adam.spitz@sun.com.

4. Running Self (detailed instructions)

On a Macintosh

The Mac OS X version of the Using_Self package includes an AppleScript application that simplifies starting up Self. You can drop a Self snapshot over the SelfDroplet application; this will create a new Terminal window which starts the Self application.

You can also associate Self snapshots with the droplet, so that double-clicking a snapshot will start it up. To do that, bring up a Show Info on any of the provided snapshots. Then in the Open

With Application section, select the SelfDroplet application in `${SELF_WORKING_DIR}/objects`.

The snapshots included with this release have their file type and creator code set correctly, while older snapshots may not. If you create new snapshots, these two properties are set correctly under Mac OS X.

You can also start Self from the Terminal. The Self VM binary is located in `${SELF_WORKING_DIR}/objects/Self.app/Contents/MacOS/Self`. To start up a snapshot, type this command:

```
<path to Self.app>/Contents/MacOS/Self -s <path to a snapshot>
```

To run Self without loading in any snapshot, just use:

```
<path to Self.app>/Contents/MacOS/Self
```

In order for the Self UI2 window to be selectable, you must have Self.app/Contents/MacOS/Self in your command. For instance, if you just type "Self" the window will not be selectable.

Running the Morphic (UI2) user interface in Mac OS X

If you are running a snapshot in which the Self user interface is not already open, you can start it from the Self prompt by typing `desktop open` (to start up Self in Carbon) or `desktop openOnDisplay: ':0'` (to start up Self in X; you can use a different display string if you wish to start up Self on an X display on another machine).

Although UI2 was originally designed for a three-button mouse, the Mac has just a one-button mouse. So, option-click on the Mac is used for a middle-button click, and command-click is used for a right-button-click. (To change these, find the `whichButton:` method in the initialization category in the `traits ui2MacEvent` object.)

On a Sun machine

The Self VM binary is located in `${SELF_WORKING_DIR}/objects`. To start up a snapshot, go to that directory and type this command:

```
./Self -s <path to a snapshot>
```

To run Self without loading in any snapshot, just use:

```
./Self
```

If you are running a snapshot in which the Self Morphic user interface (UI2) is not already open, you can start it from the Self prompt by typing `desktop openOnDisplay: ':0'.` (You can use a different display string if you wish to start up Self on a different display.)

5. Building the Self VM

This section only applies to people interested in building the Self VM. Many users will not need to do this; they can use the provided Self VM binaries. Those interested in working with the VM will need to know how to build the VM, and this section documents this process.

Common

1. You need a top-level Self directory. In your shell's start-up script, set the environment variable `SELF_WORKING_DIR` to be this directory. As an example, we use `~/self`. If you do not use a C-shell, you must still set the `SELF_WORKING_DIR` variable in your `~/.cshrc`. This is necessary since the build-time scripts are C-shell scripts and they depend on the `SELF_WORKING_DIR` variable. So as an example, if you use the bash shell, and you set `SELF_WORKING_DIR` in your `~/.bashrc`, you still need to set `SELF_WORKING_DIR` in your `~/.cshrc`.
2. Download the Working_with_VM package. We provide two package formats: a tarball and a Mac OS X disk image. The contents of the two archives are identical.
3. Extracts the contents of the Working_with_VM package.
4. Put the contents of the `put_contents_in_Self_dir` in `${SELF_WORKING_DIR}`, replacing any files if need be.
5. Put the contents of the `put_contents_in_objects_dir` in `${SELF_WORKING_DIR}/objects`, replacing any files if need be.
6. You can now delete the Working_with_VM package.

Solaris/SPARC™

First, follow the instructions in the above "Common" subsection.

Next, generate the dependency lists as follows:

1. `cd ${SELF_WORKING_DIR}/vm/svr4/generated`
2. `make lists`

The above step has to be done any time you add or delete a file to the project.

Now, you can start the regular build process:

1. `cd ${SELF_WORKING_DIR}/vm/svr4/optimized`
2. `make`
3. On a multi-processor machine, you can use parallel builds by issuing the command `make files` for the second and subsequent build jobs.

In the above instructions, I went into the optimized directory and this caused the resultant build to be an optimized build (which still included debugging information). If you want to a debug or

profiled build, go into the `debug` or `profiled directories` and issue the make command there.

Mac OS X

First, follow the instructions in the above "Common" section.

Although Self uses Carbon, you still need two X libraries. These libraries are needed to allow Self to share windows with other X clients. Also, you can use X windows, rather than Carbon windows, under OS X (see section 4). The two libraries you need are libX11.a and libXext.a (notice these are the static versions). You can use the libraries provided by the XFree86 release for Mac OS X (known as XDarwin), or Apple's own version of XFree86 (known as X11App). The XDarwin package is downloadable at http://www.xdarwin.org, and Apple's X11App is available at http://www.apple.com/downloads/macosx/apple/x11formacosx.html. Our build process expects libX11.a and libXext.a to be located in /usr/X11R6/lib. (Be careful, though, if you want to build a Self executable that can run on machines where X is not installed. If the /usr/X11R6 directory also includes the dynamic (.dylib) versions of the X libraries, Xcode will create a Self executable that links to X dynamically instead of statically. As a workaround, you can temporarily remove the .dylib files, so that Xcode will only be able to find the .a files.)

Building the Self VM under Mac OS X is now done almost completely in Xcode. However, you must first create the dependency lists in a console:

1. `cd ${SELF_WORKING_DIR}/vm/mac_osx/generated`
2. `make lists`

The above step has to be done any time you add or delete a file to the project.

Now, start up Apple's Xcode. Open the Self project file, which is in `${SELF_WORKING_DIR}/vm/mac_osx/vm_project`. Make sure that the active target is set to Self. Also, choose a build style. (Optimized Self is probably the build style you want. We don't use the Development or Deployment build styles.)

Click the Build button to start the build.

Unless you tell Xcode otherwise, the result of the build will be placed in `${SELF_WORKING_DIR}/objects/Self.app`.

Please see section 4 for instructions on how to launch the Self VM under OS X.

Porting the Self VM

Since Self 4.0, we have added a portability framework to the virtual machine and partially ported it to Mac OS X and the PowerPC architecture. The port is not quite complete because it omits the frame conversion code used to create optimized frames from unoptimized frames, and the deferred compilation of uncommon cases optimization in the optimizing compiler.

With the portability framework in place, it should be easier for others to port the virtual machine. The portability framework consists of additions to `makeDeps`, a tool that manages C++ header files, and of idioms employed when writing partially- or fully- machine-dependent classes. (See the included Self 4.1.4 Release Notes document for information on `makeDeps`.)

6. Limitations

Not to belabor the point, but the Self Intel port allows Self to run in Mac OS X on the new Intel-based Macintosh computers, but does not yet allow Self to run in Windows or Linux. We would be thrilled if someone were to get this latest release of Self working on Windows or Linux.

The Self Macintosh port is not fully integrated with Mac OS X. We would have liked the Self application to be a true Cocoa application, but time and resource limits prevent us from accomplishing this.

7. Bugs

A few really upsetting bugs are left:

The Intel version of the virtual machine crashes once in a while.

The Intel version of the virtual machine freezes up UI2 after a few hours of work. Typing "_Flush. desktop go" to the terminal window unfreezes it.

The following minor bugs exist:

In UI2, when dragging a non-rectangular object (e.g. the trash can) the shadow does not conform to the object's shape.

On the Macintosh, the spy eventually decides to paint itself over the top of the screen, no matter where it was. Of course, if you don't start the spy there is no problem.

Also on the Macintosh, sometimes the VM will complain about the task_info or thread_info call failing. This failure does not usually cause any harm. Once in a great while, though, it seems to crash Self.

If you change a module and attempt to file it out, the transporter assumes that the application is located in a folder called `objects` and assumes that the subfolders that originally contained the source file are present.

Finally, there are several tasks yet to be done that are on our wish list:

UI1 is a user-interface that was built to show how cartoon animation techniques could enhance the user's experience. It has been revived in this release, and can be run in X on the Macintosh, but does not yet run in Carbon. Also, it would be nice if the cartoon animation techniques could be incorporated into UI2 (Morphic).

When starting a snapshot on the Mac, if the screen size has shrunk, UI2 should ensure that your windows do not open outside the bounds of the new screen.

Although you can double-click a Self source file such as `all2.self` to start Self and read the file, it does not work if Self is already running.

Porting UI2 to Cocoa, the native API layer for OS X.

Finally, the icons were sketched very quickly and could really use a cleanup.

8. Conclusion

As you have read, this release embodies a lot of work with a lot of rough edges. We hope it works for you.

9. Appendix A: Detailed change log

Since Self 4.2.1:

Platform-independent changes & bug fixes:

- Revived the original Self user interface ("UI1"), which was built to show how cartoon animation techniques could enhance the user's experience. These cartoon animation techniques have not yet been integrated into the default Morphic user interface ("UI2"), but you can see UI1 if you have an X server. On a Macintosh, you will need to start up the X server, go into X's Preferences, change it to 256 color mode, and then restart the X server. Then start Self, and type the expression `ui startOn: ':0'` to start up UI1.
- Implemented David Ungar and Kristen McIntyre's patented (U.S. patent #6842182) salience-based highlighting.
- Created a morph to graphically display the results of running the performance profiler. (Call profileSlice on a block to invoke it.) It includes a pruning algorithm which attempts to eliminate uninteresting nodes from the profile tree. (You can fiddle with the settings by pressing the Controls button on the profile morph.) Both the user interface and the pruning algorithm are still very rough; we haven't had a lot of time to spend on them. If you have any suggestions or comments, please e-mail alex.ausch@gmail.com and adam.spitz@sun.com.
- Added a Find Missing Slots menu item to object outliners. This displays a list of messages that the object (or any of its parents) sends to itself, but which are not yet implemented. Thanks to the guys who did the Traits work in Smalltalk (Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew P. Black) - we got the idea from seeing their implementation in Squeak.
- Added an asSmallIntegerIfPossible method to traits bigInt, traits int32or64, and traits number. Changed bigInteger arithmetic to coerce the result to a smallInteger if it fits.
- Added a findFirst:IfPresent:IfAbsent: method to traits integer.
- Added the constant e (2.71828) to globals.

- Added an asFloat method to strings.
- Added percentile: and median methods to collections.
- Added an occurrencesOfEachElement method to collections.
- Changed findFirst:IfPresent: to return self in the absent case, rather than being an error.
- Added insertion methods to traits list (insert:AfterElementSatisfying:IfAbsent:, insert:BeforeElementSatisfying:IfAbsent:, insertAll:AfterElementSatisfying:IfAbsent:, insertAll:BeforeElementSatisfying:IfAbsent:).
- Added removeFirstIfAbsent: and removeLastIfAbsent: methods to traits list.
- Added copyWithoutPrefix: and copyWithoutSuffix: methods to traits indexable.
- Created new kinds of collections: universalSet and universalDictionary. (Ordinary sets and dictionaries use an emptyMarker and a removedMarker to indicate the absence of a key. This means that those markers cannot be used as keys in an ordinary set or dictionary. A universalSet or universalDictionary is capable of holding those markers, though it's a bit slower than an ordinary set or dictionary.)
- Created new kinds of collections: customizableSet and customizableDictionary, which can use any test (not just = like regular sets and dictionaries) to determine equality of elements.
- Added an isReflecteeEmptyBlock method to mirrors.
- Added 'IfFail:' blocks to various methods in the reflection system (like moduleInfoIfFail:), to allow the UI to be more robust.
- Added a lexicalParent fake slot to block method mirrors.
- Factored the mirror system so that all calls to primitives (e.g. _MirrorNames) go through a reflectionPrimitives object.
- Added an allTimes method to traits block, which runs the block and returns an object containing four different kinds of timing information (cpuTime, realTime, systemTime, userTime).
- Created localAccessFinder, a bytecodeInterpreter to find all local-variable-access bytecodes.
- Increased defaultProcessSize.
- In userQueryMorph, added support for nested showEverybody:While: calls.
- In defaultBehavior's error: method, trim the error string if it is too long, because otherwise the system crashes trying to draw the morphs.
- Limited the debugger's status string's length, as well.
- Fixed a bug that was causing the More Stack button in the debugger to keep appearing even when there was no more stack.
- Made the lines of the list of slots (when displaying a method) wrap if they get too long.
- Fixed a bug that was making fontStruct-caching not work.
- In traits mirrors method, made doLexicalScopes:IfFail: do a better job of respecting the fail-block.
- Made the reflection system do a better job of respecting multiple bytecode sets.
- Added a few methods to the traits object for module objects: allSubparts, sourceLineCount, sourceLineCountInMeAndAllSubparts, and slotsInMeAndAllSubparts.
- Improved the performance of searching for senders.
- Updated the VM so that it compiles under GCC 4.
- Changed the VM to use a larger new generation, and to better manage the heap when using a larger new generation.
- Fixed the VM so that defining FAST_FLOATS works again to get floats with more range.
- Fixed a bug that would cause the VM to crash in the scavenger after garbage-collection.
- Fixed a bug related to running out of space when scavenging.

- Fixed a bug with integer division.
- Added various endian primitives for byteVectors to integers.
- Fixed a few bugs in the performance profiler.
- Added a cutoff argument to the _ProfileCopyGraph primitive.
- Silenced VM complaints about bytecode position table parser errors.

Mac changes:

- Ported the Self VM to the Intel platform. Self now runs on Intel-based Macintoshes (though not yet on Windows or Linux).
- Updated the VM so that it compiles under Mac OS 10.4, a.k.a. Tiger.
- Added experimental pixel-retrieval primitives to the Carbon interface.
- Attempted to make the Mach timer more robust.

Since Self 4.2:

Platform-independent changes & bug fixes:

- The VM has been changed to work with gcc 3.3. One side-effect is that it will read old snapshots, but will save them in a format that requires the new VM.
- A bug that could cause snapshots to be unrunnable (because of specious memory exhaustion) has been fixed.
- A bug that could cause specious memory exhaustion when allocating a large array has been fixed.
- The Self-level profiler (i.e. "[10 factorial] profile") has been ported and revived. This is a hierarchical, source-level profiler.
- Keyboard handling has been made more uniform across platforms. (In particular, there were many keyboard commands that did not work in X on the Mac, and now they do.)
- New functionality has been added to the text editor: Ctrl-arrow or Option-arrow for moving by words, Shift for extending the selection. New keyboard shortcuts have been added for senders, implementors, find slot, methods containing, and "get it." The factory window has a list of all the keyboard commands.
- The mechanism for dealing with invalid slot objects has been made more robust, and now works with fake slots as well.
- Two new kinds of collections have been added: reflectiveIdentityDictionary and reflectiveIdentitySet, which compare objects using their reflective identity (rather than using their = method, as ordinary dictionaries and sets do).
- orderedSet has been made more efficient.
- The "fast accessing" methods for sets (if:IsPresentDo:IfAbsentPut:AndDo: and kin) now work even if the value being Put is not equal to the key (albeit at the cost of another hash lookup).
- The "methods containing" operation now uses a new-style slice outliner by default.
- Added support for applying a userDefinedOperation to category names as well as slots.
- Arrows between outliners now get updates. Previously, if the value of a slot changed, the arrow coming from that slot would still point to the old value.
- userProfile has been changed to use the display's host name as the user's name. (Previously, it was attempting to use the "finger" command to figure out the real name of the user, but it was

not parsing the finger output correctly. The finger code is still there, but is not being called anymore.)
- When reopening a world, Self will attempt to use the same display that the world was saved on. (So, for example, on the Mac, if you save an image in X and then quit and restart Self, the world will reopen in X, not in Carbon.)
- New core methods:
  - defaultBehavior - isNil, isNotNil, ifNil:, ifNotNil:, ifNil:IfNotNil:, ifNotNil:IfNil:
  - traits collection - allSatisfy:, anySatisfy:, noneSatisfy:, soleElement, ifNone:IfOne:IfMany:, gather:, do:SeparatedBy:
  - traits indexable - replaceFrom:UpTo:With:
  - traits string - isWordSeparator, asTokensSeparatedByCharactersSatisfying:, asWords. Also, asTokens has been renamed to asTokensSeparatedByWhiteSpace, and asTokensSeparators: has been renamed to asTokensSeparatedByCharactersIn:
  - traits abstractSetOrDictionary - at:IfAbsentPut:
  - traits pair - distanceTo:
  - browse - methods for browsing lists of the descendants of an object (not just children, but also grandchildren, great-grandchildren, and so on)
  - traits mirrors abstractMirror - everyMessageReflecteeUnderstands
  - abstract_OS - outputOfCommand:Delay:IfFail:, withTempFileDo:
  - traits random - pointInBoundingBox:

Mac changes & bugs:

- Several bugs that caused crashes (especially under Panther) have been fixed.
- Senders, implementors, etc. have been optimized with AltiVec™ instructions.
- The VM is now built under Apple's Xcode™.
- A Mach facility (task_info and thread_info) is now used to obtain process times.
- The facilities for running the Self VM under gprof have been ported to the Mac.
- The Self droplet no longer requires the SELF_WORKING_DIR environment variable to be set. However, the droplet must now be located in the same directory as the VM and the snapshot. (We would like to remove this limitation. If you would like to help and are familiar with AppleScript, please contact adam.spitz@sun.com.)

X bugs:

- We have fixed a bug that occasionally caused the xFillPolygon:GC:Xs:Ys:Shape:Mode: primitive to fail.
- We have fixed a bug that often caused copy-and-paste in X to be four bytes off (four gibberish characters at the beginning of the pasted string, and four missing characters at the end).

Since Self 4.1.6:

Platform-independent changes & bugs:

- We have further refactored the optimizing compiler, during the PowerPC SIC-porting effort. This should make it simpler for others to port the optimizing compiler to new architectures.

- For SIC development and/or testing, we have added _CompileWithSICNames primitive. If you set _CompileWithSICNames to be a vector, then only methods whose selector matches a selector in CompileWithSICNames will be compiled with the SIC.

Mac OS X changes:

- The biggest change, of course, is the availability of the SIC on PowerPC. We have noticed a substantial speedup in the Self system. Its performance is now on-par with Self on Solaris.
- The signal handling code on Mac OS X has been improved. In particular, a bug in Mac OS X causes the whole operating system to crash if a process receives a timer signal during core dump. Our workaround is to disable timer signals before core dumping.
- The build process has slightly changed. See section 5 for details.

Since Self 4.1.5:

Platform-independent bugs:

- We have fixed a virtual machine bug that caused crashes on Solaris and infinite CPU and memory consumption on the Mac. This bug would occur most frequently when accepting a change to a method in the debugger.

Mac OS X changes:

- Polymorphic inline caches (PIC) are available on the PowerPC now! This leads to a performance increase of anywhere from a factor of 0 to 3, depending on the nature of the program. We have felt the system to be more responsive. (Thank you, Michael! --Dave).
- The Self Droplet now correctly handles filenames in multiple volumes.
- Many Mac and Solaris key bindings have been implemented. In particular, command+s saves a snapshot, the arrow keys now work, and escape now does a cancel. A full list of key bindings is available in the Programmer's Environment manual.

Since Self 4.1.4:

Platform-independent changes:

- The demo snapshot has been recreated. It now reflects the most recent Self system.
- The Self debugger has been generalized to be retargetable to debug other things (for example, machine-level debugging).
- The transporter has been enhanced to handle object vectors that aren't copies of the prototypical object vector.
- We have implemented an experimental model for slots that hold methods. Under the traditional semantics, a method may only be stored in a constant slot, and the method is invoked whenever the slot is accessed. By executing `_NakedMethods: true`, the user can now select a relaxed model, which permits a method to be stored in an assignable slot. Accessing such a slot merely returns a reference to the method object.

- Added an argument count bytecode. A new instruction set has been defined and there is backward compatibility with the old instruction set. We will be placing information on this bytecode, along with the other bytecodes, on our web site soon (http://research.sun.com/self).
- `bigInt` now uses a binary base rather than a decimal base, so we can do bitwise operations.
- Support for handling 32 bit binary integers has been increased (see the `int32` and `int64` objects).
- Variations for transparent forwarding (useful for debugging) have been added (see `loggingSender`).
- `traits smallInt` has two extra methods: `numberOfOnes` and `roundUpTo:`.
- Fixed some memory leak bugs that would cause the system to slowly get bigger and bigger. (As a side effect, we have increased the time required to do a thorough memory cleanup.)
- Added support for the `ptrace` system call.
- Added a "Yank Out Outliner" menu item which lets the user create smaller views of the same object. (Warning: there are still some rough edges here.)
- Fixed bugs for frame buffers without color maps.

Solaris/SPARC™ changes:

- We wrote a new section explaining how to build the Self VM on the Solaris environment. Please see section 5.
- An inlining bug in SIC has been fixed.

Mac OS X changes:

- The Self VM is now an Apple Project Builder (PB) project. We provide the PB project, under vm/mac_osx/vm_project. Please see section 5 for instructions on building and running Self on Mac OS X.
- An AppleScript droplet has been included that facilitates starting up Self. Now, Self snapshots can be double-clicked to launch Self or they can be dropped on this droplet. Please see section 4 for information on how to start up Self on Mac OS X.
- The creator type and file type of snapshots are now set correctly (to Self and Snap, respectively).
- Minor performance enhancements.
- Aligned memory allocation doesn't use `mmap` anymore. We use `malloc` instead.
- Fixed a semaphore bug that caused the system to freeze during garbage collection.
- Added support for cursor teleportation. Thanks to Kristen McIntyre.

Since Self 4.1.3:

- The slice outliner now provides more flexible senders, implementors, etc. views of your objects.
- Several memory leaks and Macintosh performance bugs have been fixed.
- The abstract syntax module had been deleted.
- The outliner has been refactored into a pluggable outliner hierarchy and a model hierarchy. This change would facilitate supporting other languages with the Self programming environment.

- When changing a method in the debugger, if the method is present in more than one object, the debugger will ask what you want to do rather than changing it everywhere as before.
- Activation mirror objects now also contain a reference to the relevant process, simplifying their usage.
- Browsing implementors, senders, etc. is now more versatile: slice outliners have been added. These can browse hierarchically, restricting the search to an object's ancestors, descendants, or both (its family). When used in such a fashion, the results are displayed in a visual containment hierarchy corresponding to the objects' inheritance relationships.
- Some of the methods that were in outliner objects, but were specific to modifying Self objects have been relocated to mirror and slot objects.
- Outliners how have a "yank" menu button. This operation allows you to extract a view on just a single method or category, like "spawn" in Smalltalk-80.
- I have written a general parsing framework, and written a Java and Self parser in it. (As of this writing (8/5/2001), I'm not sure whether the parser will make it out the door.)
- A slot in the prototype module object, "comment", has been renamed to "myComment" in order to support robust interactive editing of the comment. (The module outliner has been revamped and now supports comment display and editing.) If you have any Self source files of your own, you should use a text editor to rename the "comment" slot in the module before reading them in.
- In order to recategorize a slot in a given object, you should now "copy" the slot and drop the copy into the new category. The system will notice that the slot is merely being recategorize and not ask for confirmation.
- A cache is now maintained of those modules that are not contained in any other modules, that is the "top-level" modules.
- Several bugs have been fixed, including one that made it impossible to single-step a thread under certain circumstances.
- OS X saves moderately-sized snapshots so quickly that I have removed the garbage collection operation before a snapshot. You can always select "clean up memory" from the background menu.The virtual machine file name suffixes have been changed to be more compatible with modern C++ compilers.
- Some of the icons have been replaced with much nicer ones (thanks to Kristen McIntyre).

Since Self 4.1.2

- Self now runs on OS X.
- A bug in the SPARC spy has been fixed.