

REPORT ON OpenDreamKit DELIVERABLE D3.7

One-click install SAGE distribution for Windows with Cygwin 32bits and 64bits

ERIK BRAY

REPORT EDITED BY LUCA DE FEO



Due on	31/08/2017 (M24)
Delivered on	29/06/2018
Lead	Université Paris-Sud (UPSud)
Progress on and finalization of this deliverable has been tracked publicly at: https://github.com/OpenDreamKit/OpenDreamKit/issues/66	

1. STATUS REPORT

One of the main tasks of WP3 is concerned with portability of the software components of OpenDreamKit. Most components are developed in UNIX environments (Linux and/or OS X), and porting them to other platforms, most importantly the Windows operating system, is usually a challenge.

Although UNIX-like systems are popular among open source software developers and some academics, the desktop and laptop market share of Windows computers is estimated to be more than 75% and is an important source of potential users, especially students.

Some OpenDreamKit components have been supporting Windows for a long time. However Windows support for SageMath, the largest of them, has been elusive for more than 10 years. This is particularly challenging, not so much because of the Sage Python library (which has some, but relatively little system-specific code). Rather, the challenge is in porting all of SageMath's 150+ standard dependencies, and ensuring that they integrate well on Windows, with full Sage's test suite fully passing.

Thus, finally bringing Windows support to the SageMath distribution has the added benefit of producing working Windows versions of all the CAS's and other software SageMath depends on, such as GAP, Singular, etc.

We are happy to report that, thanks to OpenDreamKit's efforts, and in particular thanks to Erik Bray's work at Université Paris-Sud, starting from version 8.0 of SageMath, released on July 21, 2017, a native Windows 64 bit installer based on Cygwin has been made available for all users to download¹. This is now the recommended way to install SageMath on Windows platforms.

Originally, the deliverable also had the goal of delivering an installer for Windows 32-bit based on Cygwin. Unfortunately, this goal was not achieved due to technical obstacles, that we shall briefly explain at the end of this deliverable. Fortunately, the market share for Windows 32-bit is ever shrinking, and we estimate that the lack of a 32-bit installer has a very low impact on SageMath's user base.

1.1. Description of the achievements

As of SageMath version 8.0, Sage is available for 64-bit versions of Windows 7 and up. It can be downloaded through the SageMath website, and up-to-date installation instructions are being developed at the SageMath wiki².

¹<https://www.sagemath.org/download-windows.html>

²<https://wiki.sagemath.org/SageWindows>

The installer contains all software and documentation making up the standard Sage distribution, all libraries needed for Cygwin support, a Bash shell, numerous standard UNIX command-line utilities, and the MinTTY terminal emulator, which is generally more user-friendly and better suited for Cygwin/UNIX software than the standard Windows console.

It is distributed in the form of a single-file executable installer, with a familiar install wizard interface. The download size of the installer comes in at just under a gigabyte, but unpacks to more than 4.5 GB in version 8.0.

Because of the large number of files comprising the complete SageMath distribution, and the heavy compression of the installer, installation can take a fair amount of time even on a recent system. On an Intel i7 laptop it takes about ten minutes, but results will vary. Fortunately, after nearly a year of field testing, this has not yet been a source of complaints. In future updates to the installer we may also be able to reduce the installation time/size by making certain large components (e.g. offline documentation) optional.

The installer includes three desktop and/or start menu shortcuts:

- 1) The shortcut titled just “SageMath 8.0” launches the standard Sage command prompt in a text-based console. In general it integrates well enough with the Windows shell to launch files with the default viewer for those file types. For example, plots are saved to files and displayed automatically with the default image viewer registered on the computer.
- 2) “SageMath Shell” runs a Bash shell with the environment set up to run software in the Sage distribution. More advanced users, or users who wish to directly use other software included in the Sage distribution (e.g. GAP, Singular) without going through the Sage interface.
- 3) “SageMath Notebook” starts a Jupyter Notebook server with Sage configured as the default kernel and, where possible, opens the Notebook interface in the user’s browser.

1.2. Technical details

The main challenge with porting Sage to Windows/Cygwin has relatively little to do with the Sage library itself, which is written almost entirely in Python/Cython and involves relatively few system interfaces. Rather, most of the effort has gone into build and portability issues with Sage’s more than 150 dependencies.

The majority of issues have been build-related issues. Runtime issues are less common, as many of Sage’s dependencies are primarily mathematical, numerical code. Another reason is that, although there are some anomalous cases, Cygwin’s emulation of POSIX interfaces is good enough that most existing code just works as-is. However, because applications built in Cygwin are native Windows applications and DLLs, there are Windows-specific subtleties that come up when building some non-trivial software. So most of the challenge has been getting all of Sage’s dependencies building cleanly on Cygwin, and then maintaining that support (as the maintainers of most of these dependencies are not themselves testing against Cygwin regularly).

In fact, maintenance was the most difficult aspect of the Cygwin port (and this is one of the main reasons past efforts failed—without a sustained effort it was not possible to keep up with the pace of Sage development). The critical component that was missing for creating a sustainable Cygwin port of Sage was continuous integration on Cygwin. The Sage developers maintain a flotilla of so-called “patchbots”, computers running a number of different OS and hardware platforms that perform continuous integration testing of all proposed software changes to Sage. The patchbots are able to catch changes that break Sage before they are merged into the main development branch. Without a patchbot testing changes on Cygwin, there was no way to stop changes from being merged that broke Cygwin. With some effort Erik Bray managed to get a Windows VM with Cygwin running reliably on Université Paris-Sud’s OpenStack infrastructure, that could run a Cygwin patchbot for Sage. By continuing to monitor this patchbot the SageMath community can now receive prior warning if/when a change will break the Cygwin port. We

expect this will impact only a small number of changes—in particular those that update one of Sage’s dependencies.

In so doing we are, indirectly, providing continuous integration on Cygwin for SageMaths’s many dependencies—something most of those projects do not have the resources to do on their own. So this should be considered a service to the open source software community at large.

1.2.1. Alternatives. There are a few possible routes to supporting Sage on Windows, of which Cygwin is just one. For example, before restarting work on the Cygwin port Erik Bray experimented with a solution that would run Sage on Windows using Docker. This approach “worked”, but was still fairly clumsy and error-prone. It was showcased at the first project review in April 2017, and was advertised on SageMath’s website as a temporary alternate installation method for some time before the release of SageMath 8.0.

Another approach, which was looked at in the early efforts to port Sage to Windows, would be to get Sage and all its dependencies building with the standard Microsoft toolchain (MSVC, etc.). This would mean both porting the code to work natively on Windows, using the MSVC runtime, as well as developing build systems compatible with MSVC. There was a time when, remarkably, many of Sage’s dependencies did meet these requirements. But since then the number of dependencies has grown too much, and Sage itself become too dependent on the GNU toolchain, that this would be an almost impossible undertaking with available resources.

A middle ground between MSVC and Cygwin would be to build Sage using the MinGW toolchain, which is a port of GNU build tools (including binutils, gcc, make, autoconf, etc.) as well as some other common UNIX tools like the Bash shell to Windows. Unlike Cygwin, MinGW does not provide emulation of POSIX or Linux system APIs. This would actually be the preferred approach, and with enough time and resources it could probably work. However, it would still require a significant amount of work to port some of SageMath’s more non-trivial dependencies, such as GAP and Singular, to work on Windows without some POSIX emulation. Now that the Cygwin port has been completed, a MinGW port seems more feasible.

Finally, a note on the Windows Subsystem for Linux (WSL), which debuted shortly after the start of OpenDreamKit. The WSL is a new effort by Microsoft to allow running executables built for Linux directly on Windows, with full support from the Windows kernel for emulation of Linux system calls. Basically, it aims to provide all the functionality of Cygwin, but with full support from the kernel, and the ability to run Linux binaries directly without having to recompile them. This is a very promising development. Thus, the question is asked if Sage can run in this environment, and experiments suggest that it works, albeit imperfectly (the WSL is still under active development by Microsoft and some of its interfaces are less robust than others).

A detailed account of WSL was already given in deliverable D2.3 (September 2016). In short:

- 1) The WSL is currently only intended as a developer tool: there is no way to package Windows software for end users such that it uses the WSL transparently, and users must install Microsoft development tools in order to use WSL.
- 2) It is only available on recent updates of Windows 10—it will never be available on older Windows versions.

So to reach the most users, and provide the most hassle-free user experience, the WSL is not currently a solution. However, it may still prove useful for developers as a way to do Sage development on Windows. And in the future it may be the easiest way to install UNIX-based software on Windows as well, especially if Microsoft ever expands the scope of its supported use cases.

1.3. Port for 32 bits architectures

Until 2013 the only available version of Cygwin was for 32-bit architectures. The original goal of the deliverable was to make Cygwin-based installers both for 32- and 64-bit architectures.

After getting Sage working on 64-bit Cygwin, when it came time to test on 32-bit Cygwin we hit some significant snags.

The main problem is that 32-bit Windows applications have a user address space limited to just 2 GB. This is in fact not enough to fit all of SageMath into memory at once. With some care, such as reserving address space for the most likely to be used (especially simultaneously) libraries in Sage, we can work around this problem for the average user. But the result may still not be 100% stable.

It becomes a valid question whether this is worth the effort. There are unfortunately few publicly available statistics on the current market share of 64-bit versus 32-bit Windows versions among desktop users. Very few new desktops and laptops sold anymore to the consumer market include 32-bit OSes, but it is still not too uncommon to find on some older, lower-end laptops. In particular, some laptops sold not too long ago with Windows 7 were 32-bit. According to Net Market Share³, as of writing Windows 7 still makes up nearly 50% of all desktop operating system installments. This still does not tell us about 32-bit versus 64-bit. The popular (12.5 million concurrent users) Steam PC gaming platform publishes the results of their usage statistics survey⁴, which as of writing shows barely over 5% of users with 32-bit versions of Windows. However, computer gamers are not likely to be representative of the overall market, being more likely to upgrade their software and hardware.

Given the lack of demand for 32-bit versions of SageMath on Windows, the continually shrinking market share of such systems, and the availability of legacy installation methods for them, we have decided not to pursue this effort further. Should a serious use case for such a port eventually arise, we may have to reconsider our decision.

1.4. Conclusion

Focusing on Cygwin for porting Sage to Windows was definitely the right way to go. It took only a few months in the summer of 2016 to get the vast majority of the work done. The rest was just a question of keeping up with changes to Sage and fixing more bugs. Now, however, enough issues have been addressed that the Windows version has remained fairly stable, even in the face of ongoing updates to Sage.

Porting more of Sage's dependencies to build with MinGW and without Cygwin might still be a worthwhile effort, as Cygwin adds some overhead in a few areas, but if we had started with that it would have been too much effort.

In the near future, however, the priority needs to be improvements to user experience of the Windows Installer. In particular, a better solution is needed for installing Sage's optional packages on Windows. And an improved experience for using Sage in the Jupyter Notebook, such that the Notebook server can run in the background as a Windows Service, would be desirable. This feature would not be specific to Sage either, and could benefit all users of the Jupyter Notebook on Windows.

Disclaimer: this report, together with its annexes and the reports for the earlier deliverables, is self contained for auditing and reviewing purposes. Hyperlinks to external resources are meant as a convenience for casual readers wishing to follow our progress; such links have been checked for correctness at the time of submission of the deliverable, but there is no guarantee implied that they will remain valid.

³<https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomid=0>

⁴<http://store.steampowered.com/hwsurvey/>