The Gap between Visualization Research and Visualization Software (VisGap) (2022)
C. Gillmann, M. Krone, G. Reina, T. Wischgoll (Guest Editors)

# Personal Experiences of Providing and Using Research Prototypes

Tobias Isenberg

Université Paris-Saclay, CNRS, Inria, LISN, France

**Abstract**

*I report on my personal experiences as a student, researcher, supervisor, and collaborator about providing and using research prototype software (i. e., demos). Based on an analysis of my own research activities in computer graphics and visualization, I discuss problems of providing demo software for our own projects, problems of running such software years after the release, and problems of accessing such prototypes after several years. I conclude that both source code and demos should be encouraged, provide some recommendations on how to do the latter, and call for a more active support of sharing this part of a scientific contribution within the Open Science movement.*

## 1. Introduction: The Importance of Demos and Prototypes

Demos of research prototypes can have a lasting impact.[†] This impact can be created by live or video-recorded demonstrations, but also, in particular, by *demo programs in the form of source code or executables* that allow others to independently replicate techniques. Replication is essential: it means that a demo does not only serve as an inspiration but also as a means for comparison and illustration. Future work, e. g., may want to compare a previous approach with new developments or provide overviews of different techniques. In both of these cases it may be essential to create new visual artifacts (e. g., different visual representations of the same data for comparisons or as examples) because the existing images published in scientific papers cannot be used for several reasons. One of the reasons may be that the data used in the past to create visuals is not available or not applicable for the current work. Another reason, often even more important, is that most papers including their visuals are typically covered by copyright, and it is often a tedious process to obtain permission from the copyright holders (i. e., from the publishers and not from the original authors) to re-print such work. Some researchers actually intentionally pre-record alternative images, yet more often than not this is not done or the archives are lost. In all such cases an executable demo can serve as a means to create new material.

In the past I often have been in the position to need visual examples from others for my own work, mostly when co-authoring survey articles about a given field. For example, for various research overviews (not only within the visualization field) I have used the tools Irit[‡] [Elb95], Taprats[§] [Kap00], Secord's [Sec02] stipple demo,[¶] the vIST/e project[‖] [OVVDW10], OpenWalnut[**] [EHWS10, EHS13], and brainGL[††] [BSL*14]. In other cases I have used publicly unavailable demos provided to me by the authors themselves such as the RenderBots demo [SGS05]. I sometimes asked colleagues directly to create new visuals for comparisons or as examples, such as Mould's structure-preserving stippling [Mou07, LM11] for a survey on stippling I co-authored [MARI17]. Occasionally certain techniques also get independently re-implemented by others, as demos or as part of larger tool suites. For instance, our depth-depended halos approach [EBRI09] was implemented as part of version 3 of the Unreal Engine, or the previously mentioned stippling by Secord [Sec02] has been re-implemented several times by others and made publicly available.[‡‡]

A demo can also be a powerful argument in the review process of a scientific contribution. I vividly remember that, for one of my first review assignments in a high-ranking venue in the early 2000s, I received an actual demo for a particular computer graphics approach. I was blown away at the time—I thought the authors were particularly courageous to have that much trust in their approach that they allowed the reviewers of their paper to play with their software, with the potential to discover bugs and unmentioned limitations. Nonetheless, the demo at the time was convincing and supported the written arguments in an excellent way, and as a result the respective paper submission got accepted. Taking this case as inspiration, when appropriate I also try to provide demos as part of paper submissions.

To set a good example for reproducible research, be able to provide demos for the review process, and sometimes also just for a

[†] See the "Mother of all Demos" (`wikipedia.org/wiki/The_Mother_of_All_Demos` and/or `youtu.be/yJDv-zdhzMY`).
[‡] `cs.technion.ac.il/~irit`
[§] `cgl.uwaterloo.ca/~csk/washington/taprats` (via `archive.org`)
[¶] `cs.nyu.edu/~ajsecord/stipples.html` (via `archive.org`)
[‖] `sourceforge.net/projects/viste`
[**] `openwalnut.org`
[††] `code.google.com/archive/p/braingl`
[‡‡] e. g., `github.com/azer89/Weighted_Voronoi_Stippling` or `evilmadscientist.com/2012/stipplegen-weighted-voronoi-stippling-and-tsp-paths-in-processing`

**Table 1:** *Own project websites with or without demos. Study projects not listed, but would often benefit from the study program and data being available. 33 additional project page entries not shown which do not need a demo (studies, surveys, book chapters etc.). Executables (exe) for **W**indows, unless marked for **L**inux, **M**acOS, or **iOS**). Color code: own project, supervised project (main supervisor on the respective project), participation in a collaborative project, and co-supervised collaborative project (**not** as primary supervisior, including informal co-supervisory roles). Sorted first by participation type (as color-coded) and then by year. For "(✓)" see Section 3.*

| project ID incl. year (project webpage link) | should have a demo | type of demo | sources provided | exe runs still (W) | self-hosted | special HW required |
|---|---|---|---|---|---|---|
| Strothotte1999VKV | ✓ | ✗ | ✗ | | | ✗ |
| Isenberg20003IE | ✓ | ✗ | ✗ | | | ✗ |
| Isenberg2002SSA | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Isenberg2003ADG | ✗ | exe (partial) | ✗ | ✓ | ✓ | ✗ |
| Isenberg2004CTE | ✓ | exe (partial) | ✗ | ✓ | ✓ | ✗ |
| Isenberg2006BFS | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Isenberg2006IAL | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Isenberg2008IEV | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Isenberg2008MST | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Isenberg2013VAS | ✓ | web + Java | ✗ | ✓(offline) | ✓ | ✗ |
| Isenberg2022DYB | ✓ | Python script | ✓ | ✓ | ✓ | ✗ |
| Zander2004HQH | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Isenberg2006GCS | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Schwarz2007MRP | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Grubert2008ISN | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Vlaming2008PTI | ✓ | exe | ✗ | ? | ✓ | ✓ |
| Everts2009DDH | ✓ | exe (W,L,M) | ✗ | ✓ | ✓ | ✗ |
| Svetachov2010DCI | ✓ | exe (W,L) | ✗ | ✓ | ✓ | GPU |
| Martin2010EBS | ✓ | exe (W,L) | ✗ | ✗ | ✓ | ✗ |
| Nijboer2010EFG | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Postma2010EDS | ✓ | exe (W,L) | ✗ | ✓ | ✓ | GPU |
| Yu2010FDT | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Zwan2011IMV | ✓ | exe | ✗ | ✓ | ✓ | GPU |
| Gerl2012SAI | ✓ | ✗ | ✗ | | | GPU |
| Klein2012DSD | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Yu2012ESA | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Gerl2013IEH | ✓ | ✗ | ✗ | | | ✗ |
| Everts2015EBW | ✓ | ✗ | ✗ | | | ✗ |
| Everts2015IIL | ✓ | ✗ | ✗ | | | GPU |
| Besancon2017HTT | ✓ | sources only | ✓ | n/a | ✗ | ✓ |
| Besancon2017PGF | ✓ | sources only | ✓ | n/a | ✗ | ✓ |
| Besancon2018RAR | ✓ | sources only | ✓ | n/a | ✗ | ✗ |
| Besancon2019HTT | ✓ | sources only | ✓ | n/a | ✗ | ✓ |
| Wang2019AT3 | ✓ | sources only (iOS) | ✓ | n/a | ✗ | ✓ |
| Halladjian2020SIV | ✓ | exe | ✗ | ✓ | ✓ | GPU |
| Wang2020TUA | ✓ | ✗ | ✗ | | | ✓ |
| Halladjian2022MUI | ✓ | exe | ✗ | ✓ | ✓ | GPU |
| Sonnet2003IWF | ✓ | ✗ | ✗ | | | ✗ |
| Isenberg2003IVD | ✓ | ✗ | ✗ | | | ✗ |
| Halper2003OAS | ✓ | exe (partial) | ✗ | ✓ | ✓ | ✗ |
| Jesse2003UOH | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Schmidt2007SSI | ✓ | exe | ✗ | ✓ | ✗ | ✗ |
| Stamp2007CSP | ✓ | web, Java applet | ✓ | ✗ | ✓ | ✗ |
| Neumann2007NLI | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Boukhelifa2012ESV | ✓ | ✗ | ✗ | | | ✗ |
| Wood2012SRI | ✓ | library | ✓ | ✓ | ✗ | ✗ |
| Bach2013IDG | ✓ | ✗ | ✗ | | | ✗ |
| Willet2015LRS | ✓ | web | ✓ | ✓ | ✗ | ✗ |
| Lawonn2016OBF | ✓ | ✗ | ✗ | | | GPU |
| Yu2016CEE | ✓ | exe | ✗ | ✓ | ✓ | (✓) |
| Isenberg2017VST | ✗ | dataset | n/a | n/a | ✓ | n/a |
| Isenberg2017VMC | ✗ | dataset | n/a | n/a | ✗ | n/a |
| Martin2017SDS | ✗ | exe (W,L; partial) | ✓ | ✓ | ✓ | ✗ |
| Isenberg2017VPS | ✗ | dataset | n/a | n/a | ✓ | n/a |
| Martin2019ADC | ✗ | exe (W,L) | ✗ | ✓ | ✓ | ✗ |
| Chen2021VCF | ✗ | dataset | n/a | n/a | ✗ | n/a |
| Ling2021DDR | ✓ | dataset | n/a | n/a | ✗ | n/a |
| Tietjen2005CSS | ✓ | ✗ | ✗ | | | ✗ |
| Fanea2005I3I | ✓ | ✗ | ✗ | | | ✗ |
| Meraj2008MHD | ✓ | ✗ | ✗ | | | ✗ |
| Kim2009SBE | ✓ | exe | ✗ | ✓ | ✓ | ✗ |
| Hancock2010SST | ✓ | ✗ | ✗ | | | ✓ |
| Vlaming2010I2M | ✓ | ✗ | ✗ | | | ✓ |
| Chaboissier2011RTC | ✓ | ✗ | ✗ | | | ✓ |
| Zwan2012CNN | ✓ | exe | ✗ | ✓ | ✓ | GPU |
| Kim2013BMM | ✓ | ✗ | ✗ | | | ✗ |
| Issartel2016TVP | ✓ | ✗ | ✗ | | | ✓ |
| Miao2018MVS | ✓ | ✗ | ✗ | | | GPU |
| Miao2018DDS | ✓ | ✗ | ✗ | | | GPU |
| Kouril2021HBD | ✓ | ✗ | ✗ | | | GPU |
| Lu2021CCH | ✓ | ✗ | ✗ | | | ✗ |
| Kouril2022MAN | ✓ | ✗ | ✗ | | | GPU |

personal record/backup of the demos, I thus strongly encourage my own students to provide demos for their projects if at all possible. I usually ask them to create demos for the Windows operating system, but some occasionally also produce Linux or MacOS versions (more on this choice below). To better understand my own practices for the purpose of this opinion piece, I went through all the project pages on my scientific website[†] and checked if a project should contain a demo, does contain a demo (and in what form), and if the potential demo still runs in today's environments (Table 1). In this overview I only include those projects in which a demo is meaningful, thus mostly computer graphics and visualization technique contributions. Other types of contributions such as surveys typically do not need demos, even though for a few we also collected or specifically re-implemented a subset of the surveyed techniques (e. g., [IFH*03, MARI17]). Other papers such as studies (which I also did not include in the table) are typically considered to not need a demo, yet even these could often benefit from the exact study program and the data analysis scripts being available, to facilitate replication. I color-coded the table rows by contribution type based on whether it was a project headed and implemented primarily by myself (mostly early contributions, some recent), supervised projects where I served as a main supervisor for the project, projects in which I participated as a collaborator, and finally projects where I served in some co-supervisory role (including various forms of informal co-supervision arrangements for the specific projects).

Looking at the overview in Table 1, it seems that for a good majority of projects in the first three participation types we either produced an executable demo, provided source code, or both. Only for the last type, the co-supervisions, only for 2 out of 15 projects we actually produced a demo. What this overview essentially shows is that I personally place more emphasis on providing demos compared to others. Yet we should **not point fingers** at anyone! Instead, there are several important difficulties at play that often prevent demos or source code from being made available. And I personally also did not provide demos for all projects, and in the remainder of the paper I discuss the reasons that may be responsible for why providing demos is difficult, why demos often do no longer work years later, and why many demos cease to be accessible after some time.

## 2. Problems of Providing Demos Yourself

I believe that the main reason for most people today to not provide a demo or to publish source code is that it is **extra effort**. Moreover, this effort is typically needed after the publication is accepted or the thesis is submitted—i. e., at a time when new tasks or goals have become more important. So motivating (usually) students to provide a demo at this time is difficult, in particular as they are typically **no longer paid** for the extra work and the eternal fame and glory argument may not be convincing for some.[‡] And then the demo may also be a **low priority and simply get forgotten** once the paper is accepted and published. In the past it may also have played a role that the **practice of having project pages had not been as**

[†] `tobias.isenberg.cc/Main/VideosAndDemos` (I also link to the individual project webpages from the project ID in the first column of Table 1)

[‡] Also see the discussion of journal replicability stamps and the general question of incentives in Section 5.

**established** as it is now, such as in the 2000s for myself (I only created the respective pages cited in Table 1 much later).

Another major reason that may prevent the release of a demo or source code is **problems with the licenses of used code**, such as from existing libraries or toolkits. During the actual developments these licenses often matter much less, as the code is not public and only the potential visual results are included in publications. Yet when an executable demo or even source code of the developed tool is to be published then one has to ensure that the requirements of these licenses are met. Hopefully then it does not turn out that the licenses of the used libraries are incompatible with the intended type of release, as changing to an alternative license is usually not possible at this time. Sometimes problems even arise in unexpected places: a used toolkit may allow the release of executable demos but may require that the used graphics shaders are not provided in clear text but are compiled into the demo—which the student may not have done when developing the respective demo and a recompilation may be tricky if the student has already moved on to a new job. In another example, in two of my earliest published projects [SMI99, IMS00] we had used Smalltalk as a programming environment. While it would technically be relatively easy to distribute such a project demo by sharing the project's cross-platform bytecode, at the time we used a commercial Smalltalk implementation (VisualWorks) for which it was unclear how to legally distribute the projects.[†]

Even if the student spends the extra effort and the licenses of included code are not problematic, another challenge is the need to pick a **suitable license for the distribution** of the demo or the sources. Again, the selected licenses have to be compatible with used libraries or toolkits. There are many possible choices, from a large selection of open source licenses[‡] to creative common licenses[§] to just a *"feel free to use, but at your own risk"* model. So which one is best or most suitable? A big challenge here (at least for me) is the lack of understanding of the legal implications of this choice, and if I am even at the liberty to choose a given license. This choice could be restricted by the licenses of the used libraries or even by one's employer—not to mention the potential problems if different employers or universities are involved in a publication.

The question of **license or usage rights** also involves the **included data**—which is often essential for both computer graphics and visualization. Often we may use example data from collaborators and need to make sure that we are even allowed to include this data in the demo—yet without it the demo would usually not be meaningful. And if these datasets involve information about people—such as in many visual analytics approaches—, then we also have to meet the requirements of national and international laws such as the **GDPR** in Europe.[¶] So ultimately it is not surprising that researchers often do not have the time or resources to go the extra mile to actually make a demo or source code available. Nonetheless, even if they do, there are issues with running these demos a few years after publication or even accessing them, as I discuss next.

## 3. Problems (and Solutions) for Demos to Run Years Later

From my own experience it seems that creating demos as self-contained **MS Windows™ executables** seems to work reasonably well to ensure a long survival of demos. Some of my own prototype implementations from about 20 years ago [IHS02] are still running on today's Windows versions. And for this reason I always encourage students to provide demos of their implementation at least as Windows executables, to put them on project pages.

What is problematic with this approach is the reliance on **dynamic-link libraries** (DLLs). First, it is difficult for the people who actively implement a project to actually determine which DLLs are, in fact, needed for a demo—the program obviously runs fine on their development environment because many DLLs are installed at a central location. For example, the runtime DLLs for Microsoft's Visual Studio™ are often necessary for a program to run, so it is important to test any demo on a virgin Windows installation to ensure that all needed DLLs are provided with the executable. It is not enough to state in the readme file that some Visual Studio runtime environment needs to be installed, as these may not easily be accessible years down the road.

It is also problematic to rely on **external libraries**. For example, Secord's [Sec02] stippling demo relied on an old version of the dynamically linked variant of the ImageMagick suite. Yet even such open-source projects stop providing their old binaries at some point in time. When I tried to run the Secord's stippling demo a few years ago it was close to impossible to find the old ImageMagick binaries to make it work. Newer versions also did not work because the DLL interface had changed to a degree that they were no longer compatible.[‖] In this specific case I was able to resurrect the needed binaries with the help of the Internet Archive, which happened to have backed-up some of the old installation files.[**]

Even if a demo includes all needed DLLs it may still be problematic to run them on modern versions of Windows. Either some features no longer work (as with some of my own early demos) or the executable refuses to execute some essential tasks (such as loading the data files with Secord's stippling demo). In such cases in may be helpful to resort to **virtual machines** (or old hardware) and install an old version of the operating system, which can then allow you to run the demo as expected. It can therefore be useful to keep old ISO files of Windows around.

An alternative to a one-off demo program for a single method is the implementation of the technique as part of a **comprehensive software suite**. This approach has the advantage that a larger system is likely to be supported and actively developed for a longer time, such that it can be adjusted to changing operating systems. For example, our NPR-supported sketch-based modeling [SIJ*07] has been implemented in such a larger system.[††] Within the visualization domain, similar systems that focus on a particular sub-domain are

[†] And, again, having executable project demos available on project webpages was not as common at that time as it is today.

[‡] `opensource.org/licenses`

[§] `creativecommons.org/about/cclicenses`

[¶] `gdpr.eu`

[‖] Adrian described it as "bit rot" and suggests to statically link libraries: `cs.nyu.edu:80/~ajsecord/downloads.html` (via `archive.org`).

[**] In this case it is necessary to know the correct link of the page that distributed the files, and then use `web.archive.org`'s "URLs" feature to search for the archived files for a given directory.

[††] Implemented by Ryan Schmidt in his ShapeShop (`shapeshop3d.com`).

OpenWalnut [EHWS10] and brainGL [BSL*14], or even general toolkits such as VTK [SLM04] or TTK [TFL*18].

Coming back to one-off demos, my experience is that pure **CPU realizations** survive for longer times than GPU-based implementations. Hardware extensions or GPU APIs required for a particular implementation may not be compatible between the competing GPU architectures or they may be removed from future GPUs. This removal of support also applies to or occasionally happens for CPU extensions such as Intel's recent removal of SGX, yet in my experience this occurs less frequently than for GPUs. Moreover, not all current hardware has dedicated or full-featured GPUs—many of today's laptops seem to lack such support to the degree that many of my old demos that rely on (basic) GPU support do not run on my HP laptop which is my day-to-day computing platform, despite it being equipped with an Intel GPU.

In addition to the need for GPU support, many of today's visualization approaches also rely on more than basic PC hardware. And such **proprietary special hardware** often has an active life of only a few years, such as with the touch platforms I worked on (SmartBoard [SIMC07, IEGC08] or PQLabs screen overlay [KGP*12]). Here we have both the challenge of the actual hardware to be available as well as the need for dedicated drivers (which one has to archive as well as they, too, tend to quickly disappear from the Web) to allow the operating system to interface with them. In such cases I successfully maintained the usability of our demos by insisting already at development time that the developers implemented support for common hardware such as mouse input as an alternative to other techniques such as touch sensing (a feature that can also be useful during the actual development phase). I marked such cases in Table 1 with "(✓)" to indicate that, while the project technically relies and works best on dedicated hardware, its demo still provides a way to interact with it using a generic PC setup.

In addition to this problem of dedicated hardware, a growing number of approaches requires **more than one computing device to work in sync**. This is often the case for large-screen interaction or interactive data exploration in virtual (VR) or augmented reality (AR). A demo here not only has to include the software for the actual display platform, but also for the usually mobile secondary device and, sometimes, even for a central managing server and/or a dedicated networking setup. It is often virtually impossible to provide a reproducible demo for such multi-platform setups, and in those cases my students and I have occasionally resorted to providing source code on GitHub instead of demos (even if this code is not as easy to run as executables as I had previously argued).

Connected to this issue is also the selection of architecture for **implementations for mobile devices**. Some devices may support a special type of input such as Huawei's 'force touch'[†] or Apple's '3D-touch.'[‡] Those are certainly exciting opportunities to explore as alternative means of controlling the visual exploration of data, yet with Apple's policy of closing their iOS ecosystem it makes little sense to provide an executable demo app (outside of their app store, such as on a research webpage) as it cannot easily be run by others. In the one case where we used an iOS device as the platform due to its unique sensing capabilities [WBAI19], we thus also only provided source code that would have to be deployed to a target device using `xcode`. An Android implementation, in contrast, can be run much with much less effort (as an easily installable `*.apk` file), which is why I would recommend to select Android over iOS devices for all mobile research prototypes.

One may argue that **OS-independent implementations** may be the solution to be preferred, also for PC-based prototypes. For example, one may argue that Java set out to become such an OS-independent platform. Yet in my experiences the respective APIs age faster than plain OS-dependent executables. For example, while plain Java `*.jar` 'executables' such as my own map abstraction implementation [Ise13] still run even with modern Java runtime environments (RTE) when started locally, its Web-based Java Web-start[§] no longer works due to the respective plugin no longer being supported by modern browsers. I observed the same for other demos, such as Kaplan's [Kap00] Taprats that I mentioned in Section 1.

Another recommendation to avoid incompatibility issues that I hear frequently is to use **web-based solutions**. In my view the survival rate of such demos is even lower than that of executable files, mostly because their survival often relies on few or even single people (students), after whose departure from a department the server often dies. Not even the archiving marvels of the Internet Archive's WayBackMachine help in this case, as it cannot archive the needed server infrastructure and implementation. Cipriano and Gleicher's excellent GRAPE web tool[¶] for molecular visualization [CG07], e. g., unfortunately died only a few years after having been introduced. When I tried a few years ago to use it to create visuals for an article of my own for a technique comparison I had to rely on some images they had created themselves for their own paper. Another well-known example within the visualization field—one that was even backed by a large company and not just by individuals—is Viégas et al.'s [VWvH*07] Many Eyes system, that lived from 2007 to 2015[‖] but then was retired by IBM. Java Applets that used to be a recommended way for smaller web-based implementations are also no solution [Skr20], these also no longer run even if the files are still available and one attempts to run them locally.

## 4. Problems of Provided Demos to Remain Accessible

Ultimately the question of being able to try out a research prototype, however, does not depend on whether one is able to run a specific tool on a particular hardware setup, but whether one is able to get access to the software in the first place. Even if demos of a particular approach were provided on the Web at some point, it is a question of support. Single-person initiatives (i. e., those that rely on a single or on few students and their research work) often loose support after the students graduate, and they then **quickly disappear from the Web**. Ideally one has saved a local copy of such demos in time to be able to use them later when needed (e. g., for technique comparisons or for examples for surveys). In other cases the mentioned **Internet Archive** may be able to help, as it

[†] `huawei.com/en/news/2015/09/hw_452909`

[‡] `developer.apple.com/design/human-interface-guidelines/ios/user-interaction/3d-touch`

[§] `osmabstraction.isenberg.cc`

[¶] `grape.uwbacter.org` (via `archive.org`)

[‖] `bewitched.com/manyeyes.html`

occasionally also archives the linked binaries. In such cases it helps to store at least the URLs of the project pages, which are needed to look up the stored files. For example, it can help to accompany surveys of the research literature with links to the respective tools that were used to create example images (e. g., in footnotes) as I did for my illustrative DTI visualization techniques survey [Ise15] or in this very paper. Alternatively, the same purpose can also be served by a paper appendix that contains an overview of resources for a particular topic at publication time, like we did for a survey of digital stippling [MARI17].

As I had mentioned in the introduction, in lucky cases certain approaches also get picked up by others and are either **re-implemented** as such or as part of a larger toolkit. An example is my colleague Domingo Martín's Stipple Shop tool[†] that he created as part of the same survey of digital stippling [MARI17]. Yet this is rare overall and even it if happens one cannot be sure that the re-implementation is fully faithful to the original implementation or approach. Also, the question of the long-term accessibility of the re-implementation is the same as for the original tool, if it existed.

While there are certainly examples of single people hosting tools over multiple decades,[‡] **modern source code archives** such as SourceForge and, more recently, GitHub promise to solve this hosting problem. They guarantee the availability over longer periods of time compared to self-hosted software. Moreover, they not only provide means to share source code but also support software authors with making binaries available, including hosting old releases for extended periods of time. As such they seem to be better suited for sharing one's demos, and could also be a solution for cases where sharing the source code is not an option. Nonetheless, even for large hosting services they also rely on the continued financial support of their backing companies or institutions. For example, Google's Code repository today only survives in form of a static archive.

## 5. Limitations, Summary, and Conclusion

With this opinion piece I have tried to provide an overview of my own personal experiences concerning the creation, hosting, accessing, and use of research prototypes or demos of my own work and of others. Naturally, this discussion is quite self-focused; yet the mentioned issues and at least partial solutions also should apply to the visualization field at large or other fields where demos would be helpful. Nonetheless, I want to point out again that I focused primarily on prototypical demos whose goal is to allow others try out published techniques and/or produce additional results for comparison—I have largely not been talking about the development of real domain applications, which certainly have additional points of concern to remain available, relevant, or usable.

I can thus summarize my experiences and advice as follows. I would recommend to provide both, executable demos *and* source code—both are needed yet for different forms and goals of replication. While I have not personally done the latter to a large degree for my own work yet, it seems to be increasingly supported by universities and institutions, which is great. But I also think that my approach to provide executables still is important as they allow others to replicate one's approaches more easily than if only source code is available. For such executables, I recommend Windows binaries, for which all required libraries/DLLs should be included to make it possible to run the demo at a later time on a pristine OS. If special hardware is needed, it helps potential users to provide work-arounds to make the tool usable on common hardware. When you share demos or code, it may be better in the long run to rely on established large platforms such as GitHub (and also research-focused sites such as OSF[§]), as opposed to self-hosting. Also, as a supervisor, regularly check if the respective demos are, in fact, created—potentially using an 'exit procedure' for graduating students that includes such a requirement. If a demo is not possible for some reason, then provide a video and also create additional visuals for colleagues who may need examples that are not covered by copyright other than that of the original authors (who can give permission for a later reuse by others relatively easily).

In conclusion, the question of whether source code is made available or demo programs are shared, maintained, and kept available is ultimately primarily a question of the **incentive** to overcome the described hurdles. In recent years, several journals (in our field, e. g., TVCG, CGF, TOG, C&G) have started to introduce what they call "replicability stamps."[¶] While this is a nice initiative and may provide an additional incentive, it unfortunately still only boils down to the **eternal fame and glory argument** I mentioned in Section 2.[‖] As others have discussed in detail (e. g., [Cro19]), in today's scientific publishing landscape the focus on established, easily measurable quantitative quality metrics does not encourage reproducible work, it encourages highly cited work and a high output. This leaves little time for reproducibility (in the form of source code, demos, or otherwise). **Open science practices** (e. g., [BPSS*21]) should further be encouraged and expected by default, including not only the paper itself, its data, and the respective source code but also binaries to change the incentive from 'eternal fame and glory' to a prerequisite for a paper being accepted—at least for the majority of publications. Such a requirement for paper acceptance then may also change the game overall and encourage the establishment of a default and long-supported platform(s) of demos or implementations, such that many or most of the issues I discussed become obsolete.

## Full Disclosure

In the process of the navel-gazing for this write-up, I added six more demo programs to my project pages, resulting in the summary in Table 1. Specifically, I added three old demos, one of which I found in my archives, while two others were provided to me by my former students Angela Brennecke and Xiyao Wang (王曦耀) from their own archives. The other three demos I added were for projects whose papers were just recently accepted. I also replaced another old demo with a different executable I found in my archives because the original one would only start under very specific circumstances.

[†] `github.com/dmperandres/StippleShop` and `tobias.isenberg.cc/VideosAndDemos/Martin2017SDS`

[‡] One example is Gershon Elber's Irit rendering software that I mentioned in Section 1, it has been available for more than 25 years.

[§] Center for Open Science: `osf.io` and `cos.io`.

[¶] Also see the Graphics Replicability Stamp Initiative (GRSI) `replicabilitystamp.org` and `sites.google.com/site/drminchen/cgf-info/cgf-stamp`.

[‖] A small exception is TVCG: it promises faster acceptance-to-final-publication times for articles with the stamp, compared to those without it.

## Acknowledgments

I am extremely grateful to all my past and current students and collaborators who created demos and/or provided source code, data, etc. that I am able to put on or reference from our research project pages. Thanks a lot! Also thanks to Lonni Besançon for comments, pointers to the literature, and the many discussions on the subject of open science, to Mickaël Sereno for his compiler wizardry to finish some recent demos, and to Raimund Dachselt for the idea of keeping an archive of alternative/extra images for each publication. Finally, thanks to Petra Isenberg for general comments and proof-reading.

## References

[BPSS*21] Besançon L., Peiffer-Smadja N., Segalas C., Jiang H., Masuzzo P., Smout C., Billy E., Deforet M., Leyrat C.: Open science saves lives: Lessons from the COVID-19 pandemic. *BMC Medical Research Methodology 21* (June 2021), 117:1–117:18. doi:10.1186/s12874-021-01304-y.

[BSL*14] Böttger J., Schäfer A., Lohmann G., Villringer A., Margulies D. S.: Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 471–480. doi:10.1109/TVCG.2013.114.

[CG07] Cipriano G., Gleicher M.: Molecular surface abstraction. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1608–1615. doi:10.1109/TVCG.2007.70578.

[Cro19] Crous C. J.: The darker side of quantitative academic performance metrics. *South African Journal of Science 115*, 7–8 (July 2019). doi:10.17159/sajs.2019/5785.

[EBRI09] Everts M. H., Bekker H., Roerdink J. B. T. M., Isenberg T.: Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov./Dec. 2009), 1299–1306. doi:10.1109/TVCG.2009.138.

[EHS13] Eichelbaum S., Hlawitschka M., Scheuermann G.: LineAO—Improved three-dimensional line rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (Mar. 2013), 433–445. doi:10.1109/TVCG.2012.142.

[EHWS10] Eichelbaum S., Hlawitschka M., Wiebel A., Scheuermann G.: OpenWalnut – An open-source visualization system. In *Proc. 6th High-End Visualization Workshop* (2010), Lehmanns Media—LOB.de, Berlin, pp. 67–78.

[Elb95] Elber G.: Line illustrations ∈ computer graphics. *The Visual Computer 11*, 6 (June 1995), 290–296. doi:10.1007/s003710050022.

[IEGC08] Isenberg T., Everts M. H., Grubert J., Carpendale S.: Interactive exploratory visualization of 2D vector fields. *Computer Graphics Forum 27*, 3 (May 2008), 983–990. doi:10.1111/j.1467-8659.2008.01233.x.

[IFH*03] Isenberg T., Freudenberg B., Halper N., Schlechtweg S., Strothotte T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications 23*, 4 (July/Aug. 2003), 28–37. doi:10.1109/MCG.2003.1210862.

[IHS02] Isenberg T., Halper N., Strothotte T.: Stylizing silhouettes at interactive rates: From silhouette edges to silhouette strokes. *Computer Graphics Forum 21*, 3 (Sept. 2002), 249–258. doi:10.1111/1467-8659.00584.

[IMS00] Isenberg T., Masuch M., Strothotte T.: 3D illustrative effects for animating line drawings. In *Proc. IV* (2000), IEEE Computer Society, Los Alamitos, pp. 413–418. doi:10.1109/IV.2000.859790.

[Ise13] Isenberg T.: Visual abstraction and stylisation of maps. *The Cartographic Journal 50*, 1 (Feb. 2013), 8–18. doi:10.1179/1743277412Y.0000000007.

[Ise15] Isenberg T.: A survey of illustrative visualization techniques for diffusion-weighted MRI tractography. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*, Hotz I., Schultz T., (Eds.). Springer, Berlin/Heidelberg, 2015, ch. 12, pp. 235–256. doi:10.1007/978-3-319-15090-1_12.

[Kap00] Kaplan C. S.: Computer generated Islamic star patterns. In *Proc. Bridges* (2000), Bridges Conference, pp. 105–112. URL: https://archive.bridgesmathart.org/2000/bridges2000-105.html.

[KGP*12] Klein T., Guéniat F., Pastur L., Vernier F., Isenberg T.: A design study of direct-touch interaction for exploratory 3D scientific visualization. *Computer Graphics Forum 31*, 3 (June 2012), 1225–1234. doi:10.1111/j.1467-8659.2012.03115.x.

[LM11] Li H., Mould D.: Structure-preserving stippling by priority-based error diffusion. In *Proc. Graphics Interface* (2011), CHCCS, Canada, pp. 127–134. URL: https://graphicsinterface.org/proceedings/gi2011/gi2011-17/.

[MARI17] Martín D., Arroyo G., Rodríguez A., Isenberg T.: A survey of digital stippling. *Computers & Graphics 67* (Oct. 2017), 24–44. doi:10.1016/j.cag.2017.05.001.

[Mou07] Mould D.: Stipple placement using distance in a weighted graph. In *Proc. CAe* (2007), Eurographics Association, Goslar, pp. 45–52. doi:10.2312/COMPAESTH/COMPAESTH07/045-052.

[OVVDW10] Otten R., Vilanova A., Van De Wetering H.: Illustrative white matter fiber bundles. *Computer Graphics Forum 29*, 3 (June 2010), 1013–1022. doi:10.1111/j.1467-8659.2009.01688.x.

[Sec02] Secord A.: Weighted Voronoi stippling. In *Proc. NPAR* (2002), ACM, New York, pp. 37–44. doi:10.1145/508530.508537.

[SGS05] Schlechtweg S., Germer T., Strothotte T.: RenderBots—Multi-agent systems for direct image generation. *Computer Graphics Forum 24*, 2 (June 2005), 137–148. doi:10.1111/j.1467-8659.2005.00838.x.

[SIJ*07] Schmidt R., Isenberg T., Jepp P., Singh K., Wyvill B.: Sketching, scaffolding, and inking: A visual history for interactive 3D modeling. In *Proc. NPAR* (2007), ACM, New York, pp. 23–32. doi:10.1145/1274871.1274875.

[SIMC07] Schwarz M., Isenberg T., Mason K., Carpendale S.: Modeling with rendering primitives: An interactive non-photorealistic canvas. In *Proc. NPAR* (2007), ACM, New York, pp. 15–22. doi:10.1145/1274871.1274874.

[Skr20] Skrodzki M.: How the deprecation of Java applets affected online visualization frameworks – A case study. In *Proc. VisGap* (2020), Eurographics Association, Goslar, pp. 59–67. doi:10.2312/visgap.20201111.

[SLM04] Schroeder W. J., Lorensen B., Martin K.: *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, New York, 2004. URL: https://vtk.org/vtk-textbook/.

[SMI99] Strothotte T., Masuch M., Isenberg T.: Visualizing knowledge about virtual reconstructions of ancient architecture. In *Proc. CGI* (1999), IEEE Computer Society, Los Alamitos, pp. 36–43. doi:10.1109/CGI.1999.777901.

[TFL*18] Tierny J., Favelier G., Levine J. A., Gueunet C., Michaux M.: The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan. 2018), 832–842. doi:10.1109/TVCG.2017.2743938.

[VWvH*07] Viégas F. B., Wattenberg M., van Ham F., Kriss J., McKeon M.: Many Eyes: A site for visualization at Internet scale. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1121–1128. doi:10.1109/TVCG.2007.70577.

[WBAI19] Wang X., Besançon L., Ammi M., Isenberg T.: Augmenting tactile 3D data navigation with pressure sensing. *Computer Graphics Forum 38*, 3 (June 2019), 635–647. doi:10.1111/cgf.13716.