MediaScripting Introductory Computer Science with a Free and Open-Source Graphics Application

Søren Berg, Janet Davis, Samuel A. Rebelsky
Dept. of Computer Science, Grinnell College
Grinnell, IA 50112
641-269-4410
{bergsore,davisjan,rebelsky}@grinnell.edu

ABSTRACT

Free and Open Source Software (FOSS) is used in many places in the Computer Science curriculum, from introductory courses (in which students often use open-source compilers, such as GCC, languages, such as Python, and program development environments, such as Eclipse) to mid- and upper-level software design courses, in which students contribute to open-source projects. In this position statement, we suggest a different use of FOSS in the introductory course: Since many FOSS applications now support scripting, students can learn basic concepts in computer science through scripting FOSS applications. We discuss this concept in the context of our introductory course, in which students explore image creation and manipulation as the primary problem domain, drawing upon the ideas of Media Computing [3], with programming done within the context of Script-Fu, the scripting language for The Gnu Image Manipulation Program (GIMP) [2].

Categories and Subject Descriptors

K.3.2. [**Computers and Education**]: Computer and Information Science Education – *Computer Science Education*. D.3.2. **[Programming Languages]**: Language Classifications - *Applicative (functional) languages*

General Terms

Languages, Design, Human Factors

Keywords

Free and Open-Source Software. Introductory Computer Science. Media Computing. Scripting.

1. INTRODUCTION

It all started with a simple idea: Why not show students the value of the introductory CS course in a concrete domain? We teach our introductory course in Scheme, and, while we see many benefits to teaching in Scheme, we also find that students regularly complain that they see no "real world" applications of Scheme. In early 2001, one of us decided to add a module on scripting the GIMP to the first course. Script-Fu, the scripting language of the GIMP [2], is simply a version of Scheme, and students clearly understood the importance of image editors as an area of computation. Script-Fu provided a rich environment for programming, including access not only to the pixels of an image, but also to all the tools available through the graphical user interface. Students can write programs that choose brushes, select and modify regions, drag across the screen, and more.

The initial course module was relatively small: Students spent one class period (fifty minutes) learning the GIMP, one period learning the basic Script-Fu procedures, one period writing their own procedures, and one exploring a variety of techniques for algorithmically creating images.

In the years since, we've extended this approach in many ways. We have used GIMP scripting to ground ideas of higher-order programming in an upper-division programming languages course, to introduce students to CS during orientation, as the subject of summer research projects, and, most recently, as the core of our introductory course.

Our experience suggests a number of lessons, including:

- Because of the nature of FOSS, it is possible (although not always easy) to repurpose a FOSS application to support teaching computer science.
- Scripting a FOSS application can motivate students in introductory computer science.
- Scripting a FOSS application can encourage students to participate more actively in the FOSS community.
- Repurposing can be dangerous because FOSS applications sometimes move in different directions than we would hope or expect.

We discuss these issue and others, below.

2. SCRIPTING FOSS APPLICATIONS IN INTRODUCTORY CS

At the core of this approach is the notion that application scripting is useful for introductory students. Application scripting is an important and growing paradigm for computing. As Warren [6] notes, "modern 'scripting languages' are sophisticated and powerful environments in which students can learn the basics of programming without [...] unnecessary complications".

For students who don't plan to go on in computer science, scripting provides a clearer insight into the powers of algorithmic

Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium March 4th 2009 CC: BY SA NC. Copyright 2009 Samuel A. Rebelsky, Søren Berg, and Janet Davis. Humanitarian FOSS Project

thinking. While many non-majors would not envision regularly writing applications, most can see the benefit of writing small scripts to automate tasks, provide more precise results, and otherwise give them better control of their application.

Application scripting clearly benefits from the FOSS movement. Incorporating a proprietary language, like Visual Basic, into an application requires getting permission and includes a danger that the owner of the language will discontinue support or significantly change the language. Certainly, the FOSS community has developed a number of scripting languages that can now be incorporated into applications, from Tcl/Tk to Python and beyond.

As importantly, designers of FOSS applications, often more than designers of proprietary applications, understand the power of scripting languages. For example, while it is certainly possible to extend Photoshop by developing plug-ins, doing so requires relatively deep knowledge of both Photoshop and of programming techniques. In contrast, even from its early days, the GIMP has included Script-Fu, a relatively simple scripting language. (Early GIMP documentation even included a humorously-named “Black Belt School of ScriptFu” [5].)

However, like other areas of computing, scripting has seen “language wars” between scripting languages. If scripting is the sole motivation for learning a language, these wars are dangerous, since the scripting language a student learns for one application is unlikely to be available in another application.

Fortunately, most open-source applications provide an infrastructure that supports multiple scripting languages. For example, while the GIMP initially supported only a Scheme-based scripting language, it now supports Python and Perl. Similarly, although Open Office originally emphasized Java, it now supports scripting in Python, Basic, and many other languages.

3. FOSS PERSPECTIVES

Scripting an open-source can also subtly encourage students to embrace (or at least engage with) a FOSS perspective. Certainly, students used to having to pay for special software (e.g., Minitab for Statistics, Photoshop for Digital Art) appreciate being able to use their class software on their own computers without additional cost. As importantly, because many have found that a variety of proprietary software will not run on their computers (e.g., Mac users and Minitab), students may appreciate that most FOSS software can run on most platforms.

However, scripting FOSS applications provides the “freedom” in FOSS: By scripting FOSS applications, students can come to understand that they are free to modify the application to meet their own needs, and that scripting provides the mechanism to do that. (Obviously, students may need to some prompting to understand this perspective, but it is a natural outcome.)

Finally, student scripts can be contributed back to the core applications. For example, GIMP provides a relatively open repository of Script-Fu scripts, and students can be encouraged to contribute scripts to that repository. Through such participation, students can become better attached to the FOSS community.

4. HAZARDS OF REPURPOSING

While FOSS applications often include scripting languages and, by their nature, allow us to extend the scripting environment for our class purposes, there a certainly dangers in relying on FOSS applications for this purpose. Foremost of these is the likelihood of a conflict of perspectives. For example, most of the Script-Fu documentation teaches Scheme as an imperative language, and ignores the functional attributes we try to encourage. A importantly, different perspectives significantly affect the codebase. For example, during this period, GIMP switched the underlying implementation of Script-Fu from SIOD [1], a robust and relatively language, to TinyScheme [4], a more powerful, but ultimately less reliable version. During this time, the GIMP also switched its scripting emphasis from Script-Fu to Python. Because of the switch in primary scripting language, one person, with particular goals, could drive the change to the implementation of Script-Fu.

These changes affected us in a variety of ways: First, what had been an acceptable environment for novice programmers became a bit less acceptable. Second, because a smaller team was supporting Script-Fu, questions and suggestions about the underlying implementation were sometimes met by silence. (We had written a number of extensions based on the previous implementation, and the research students rewriting them often encountered problems) It is not that questions were ignored, as there were certainly times that research students received “live” support through IRC. Rather, it’s that such support was necessary, inconsistent, and unpredictable. Third, the change in perspective meant that some questions were met with relatively derisive answers (e.g., “Why are you trying to do something in Script-Fu? The real work is going on in Python.”).

5. CLOSING NOTES

In spite of the problems mentioned in the prior section, the experience of using a FOSS application to teach introductory computer science has been primarily a positive one. Certainly, the approach has the many benefits described above. In addition, the research students who have developed extensions to FOSS applications for use in the introductory classroom generally found good support for their work, not only in terms of answers, but also in terms of encouragement. Finally, we probably could not have made the same extensions with proprietary software.

6. REFERENCES

[1] Carrett, G. J. 1996. Scheme in One Defun. http://www.cs.indiana.edu/scheme-repository/imp/siod.html

[2] The GIMP Team, 2008. GIMP - The GNU Image Manipulation Program. Online resource, available at http://www.gimp.org.

[3] Guzdial, M. 2003. A Media Computation Course for Non-Majors. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science*, pp. 104-108.

[4] Souflis, D. and Shapiro, J. (n.d.). TinyScheme. http://tinyscheme.sourceforge.net/home.html

[5] Terry, Mike. 1997. Mike Terry's Black Belt School of ScriptFu. Originally at http://www.klaban.torun.pl/help/script-fu-tutorial/. Now archived at http://www.linbox.com/ucome.rvt/any/doc_distrib/gimp-1.1.18/manual/manual/GUM/write_scriptfu3.html

[6] Warren, P. 2001. Teaching programming using scripting languages. *J. Comput. Small Coll.* 17, 2 (Dec. 2001), 205-216.