A Preview of Pencil Code

A Tool for Developing Mastery of Programming

David Bau
Google
davidbau@google.com

D. Anthony Bau
Phillips Exeter Academy
dab1998@gmail.com

Abstract

Pencil Code is an educational programming tool designed to help students overcome common obstacles to advancing to open-ended work with real world languages and libraries.

It transitions from visual code to text code with a dual-mode block and text editor; it smooths the leap from guided to open-ended work by integrating tutorials with a general-purpose programming tool; and it bridges the gap from educational functions to standard APIs by including a turtle library that subclasses jQuery.

1. Introduction

Educational programming tools generally use one of two methods:

1. They help beginners achieve satisfying results quickly in a way that minimizes frustration.
2. Or they introduce beginners directly to programming systems used by professionals.

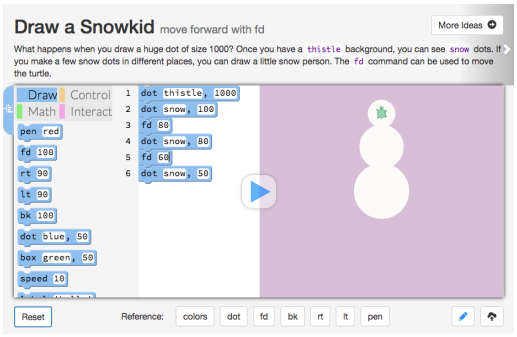
Pencil Code is designed to bridge the two styles of learning. It is welcoming to beginners, with simple touch-enabled drag-and-drop UI, tutorials with hints, and a turtle graphics canvas. At the same time, it introduces idioms used by professional web developers:

- The tutorial system ("Gym") transfers work to an IDE where students can create larger projects and build their own websites.
- Blocks in the visual programming editor are equivalent to statements in the mainstream text language CoffeeScript, and students can freely switch between text and block modes.
- The turtle graphics library is an extension of the popular library jQuery. Students exploring the limits of turtle graphics will discover that they have been using jQuery all along.

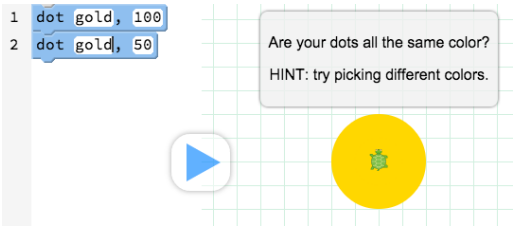
The goal of Pencil Code is to build enough confidence in beginning programmers so that they can create programs without Pencil Code.

2. From the Gym to the Web

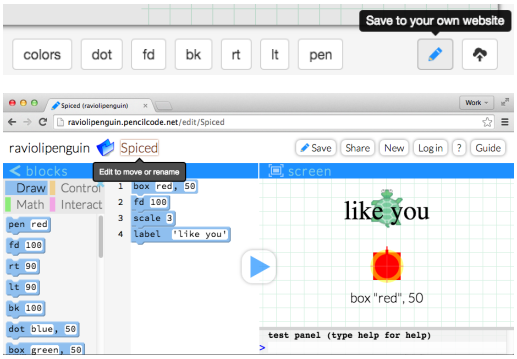
New users of Pencil Code are brought to a tutorial area called the Pencil Code Gym, which provides guidance on how to create specific projects involving turtle graphics, music, and storytelling. Projects in the gym focus attention on a few concepts at a time.



The Gym analyzes programs and shows explanations, hints, and suggestions based on the context. For example, if a program draws two dots in a way that produces a result that might be hard to understand, it shows a hint like the following:



Pencil Code is designed to help students make the transition to a less scaffolded development environment when they are ready to share their work. The "Save to your own website" button brings them to an editor that strips away the tutorial interface.



By transferring programs from the tutorial page, Pencil Code introduces an open-ended workspace while avoiding the intimidating

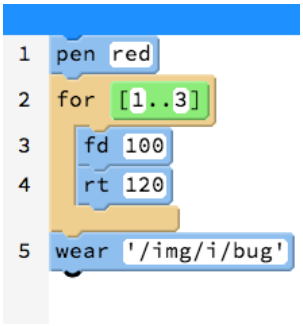
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-part components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).
SPLASH '14 Compantion October 20-24, 2014, Portland, Oregon, USA
ACM 978-1-4503-3208-8/14/10

step of facing a blank page. The general-purpose workspace allows students to build a portfolio and build more complex projects.

3. From Blocks to Code

Another hurdle for students is the transition from visual programming to using a text programming language.

(a) Scratch

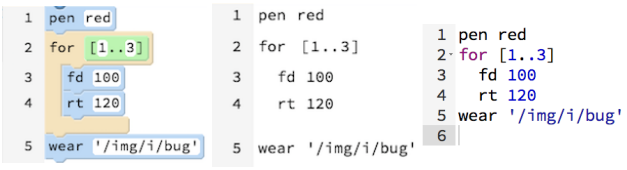


(b) Pencil Code

As illustrated above, the Pencil Code Block editor is inspired by the visual programming languages Scratch[1] and Blockly[2]. However, there are two notable differences:

- Instead of natural language text and icons, each block is labelled with function names, arguments, and syntax from the text language CoffeeScript.
- Instead of being arranged freely in two dimensions, blocks are snapped into a single main program in linear order. The program is labelled with line numbers.

These differences mean that, when a beginner is building a program with blocks, they are also learning to use a text language. The mode can be switched between blocks and text at any time.



The Pencil Code block editor is a component called Droplet, which is a new mulitlanguage editor that supports switching between blocks and text code. Droplet shows a smooth animation, illustrated above, when switching modes.

- Droplet can show any CoffeeScript program as blocks, so beginners can load any program that had originally been written text and work with it using drag-and-drop.
- The switch between blocks and text can be done freely without losing any formatting, so students can use blocks as an aid even when they are learning text programming. Blocks allow students to experiment with unfamiliar language constructs, then switch back to text to see the syntax.
- The ability to support both modes with the same programs makes it possible for students at different levels to collaborate and participate in the same classroom.

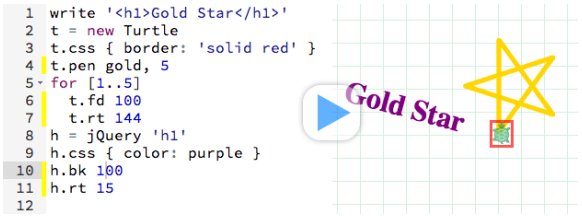
The Droplet block editor is *not* a new programming language. It is a visual code editing library that can be used as a component of any web-based code editing tool. It can support any text language given an appropriate parser adapter.

4. From Turtles to jQuery

The third hurdle addressed by Pencil Code is the gap between simplified turtle functions and professional programming idioms. Since Pencil Code uses CoffeeScript, a full-featured and popular language used by web developers, it is a good tool for learning about functions, objects, classes, closures, recursion, networking, and data structures. It is also a good tool to use to learn about mainstream programming APIs and libraries.

The most popular Javascript library used on the web is jQuery, used on more than 60%[3] of top websites in 2014. The turtle function library used by Pencil Code is designed as a jQuery extension.

In the Pencil Code turtle library, every turtle is a jQuery object, and the base jQuery functions are extended with turtle functions. That allows students to learn to use jQuery by analogy with turtles:



The program above uses the standard jQuery function `css` to add a border around a Turtle created using `new Turtle`. It uses the same function to set the color of the contents of an HTML `<h1>` element selected using `jQuery`; and it moves both objects with turtle motion.

Teaching with jQuery and CSS has several advantages:

- Popular programming technologies can motivate students because they are seen as 'authentic'.
- Learning with technologies that have a large professional community means that students can *learn to learn* using online resources like forums, tutorials, and references.
- Popular languages and libraries have direct vocational value.

5. Related Work

The tutorial structure and hints used in the Pencil Code Gym are inspired by the programming tutorials on learn.code.org[4] and Kahn Academy[5].

Because the focus of the Pencil Code Gym is to encourage students to work creatively, the Pencil Code Gym emphasizes an open-ended canvas with ideas and reference material, and the ability to publish work on the web.

Alice[6] and Scratch[1] pioneered the type of visual programming idiom used by Pencil Code's visual block editor. App Inventor[7] brought this idiom to the world of mobile application development, and Blockly[2] brought this idiom to the web with a modular Javascript and SVG-based implementation.

These visual tools have some support for text code. Scratch supports the ability to write extensions in text languages, although it does not translate between blocks and text code. App Inventor has a text language that can represent the same logic as blocks. Blockly and Alice support generation of text code from blocks, however they do not provide students with the ability to work in text and then switch to blocks.

Text-to-block translation was recently implemented in the Tiled Grace[8] project. Tiled Grace provides an animated transition between blocks and text.

The Droplet editor used in Pencil Code differs from Tiled Grace in its full support for all text syntax: for example, it preserves comments and white space when switching modes. That capability

means that students using Pencil Code can switch between blocks and text without losing any of the formatting of their work.

Several previous systems have implemented educational libraries in general-purpose professional programming languages. For example, Python's built-in turtle package[9] provides a Tk-based implementation of turtle graphics in that language. Another notable library is Processing[10], which bridges the gap between beginner and professional idioms by providing a beginner-friendly API in a Java language variant that is intended for use by creative professionals.

Previous work to bring turtle graphics to Javascript includes Papert[11], a Javascript implementation of Logo, and Turtlewax[12], a lightweight turtle library. Pencil Code's turtle library differs from these implementations because it is designed to help teach jQuery: in Pencil Code, each turtle is a jQuery object.

6. Future Work

Pencil Code is being tested in classrooms, with planned efforts to test the tool with middle-school, high-school, and college-age students. The goal of the testing is to find effective ways to apply the dual-mode text and block editor in classes; and to determine whether the tool is effective in helping attract, teach, and motivate students towards mastery of computer science.

Support for other languages is being added to Pencil Code. Work to add Javascript support is underway: Javascript support will allow students to directly apply a large body of online tutorial material. Similarly, HTML and CSS support is planned, as well as support for other widespread educational programming languages such as Java.

Pencil Code capabilities are being made available as modules: for example, work is ongoing to make the Droplet text-and-block editor usable as a module for code.org and other educational code development environments.

Interactive visual debugging capabilities are planned, to allow students to use Pencil Code to understand unseen state in their programs.

Integration with professional tools such as Github and JsFiddle are planned, to encourage students to move from Pencil Code to more flexible and powerful tools.

Acknowledgments

Thanks to Hal Abelson and the teachers of CSTA whose discussions and suggestions inspired the design of Pencil Code; to Google for its support; to the teachers and students at Dever-McCormack, Beaver Country Day, Worcester Tech, AMSA, Lincoln School, Phillips Exeter, and other schools who have done early testing; and to the open source contributors to Pencil Code and its components.

References

[1] John Maloney, Leo Burd, Yasmin Kafai, Natalie Rusk, Brian Silverman, and Mitchel Resnick. 2004. Scratch: A Sneak Preview. Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing (C5 '04), 104-109.

[2] Neil Fraser. Google blockly - a visual programming editor. URL: http://code.google.com/p/blockly. Accessed Aug. 2014.

[3] John Resig. The state of jQuery 2014. URL: http://blog.jquery.com/2014/01/13/the-state-of-jquery-2014/. Accessed Aug. 2014.

[4] Code.org. Hour of Code Tutorial. http://learn.code.org/ hoc/1. Accessed Aug. 2014.

[5] Kahn Academy. Computer Programming Tutorials. https://www.khanacademy.org/computing/cs/programming. Accessed Aug. 2014.

[6] S. Cooper, W. Dann, and R. Pausch, Teaching objects-first in introductory computer science. SIGCSE Bulletin, vol. 35, no. 1, 2003.

[7] H. Abelson. App Inventor for Android. Google Research Blog, July 2009. http://googleresearch.blogspot.com/2009/07/app-inventor-for-android.html. Accessed Aug. 2014.

[8] M. Homer, J. Noble, A tile-based editor for a textual programming language. Proceedings of the First IEEE Working Conference on Software Visualization, Sept. 2013.

[9] Gregor Lingl, New Turtle Module. Python issues database, June 2006. http://bugs.python.org/issue1513695. Retrieved Aug. 2014.

[10] Reas, Casey, and Ben Fry. 2006. "Processing: programming for the media arts." AI and Society 20.4: 526-538.

[11] Thomas Figg and contributors. 2008. "Papert: a Logo interpreter in Javascript", https://code.google.com/p/papert/. Retrieved Aug. 2014.

[12] Dave Balmer. 2010. "Turtlewax: Logo-style turtle graphics in JavaScript using HTML5 Canvas." https://github.com/davebalmer/turtlewax. Retrieved Aug. 2014.