# Experiences of Using a Collaborative Programming Editor in a First-Year Programming Course

**Troy Harding**
**Professor**
**Engineering Technology Department**
**Computer Systems Technology**
**Kansas State University Salina**

## Abstract

Recent research has demonstrated that collaborative learning can be an effective method for engaging millennial students.[1,2] This paper highlights experiences of using a collaborative editor to facilitate learning in a first-year programming course. The paper will describe how the collaborative editor was customized for the class and how it was utilized by the teacher and the students. The web-based editor allowed students to see and edit the same program file and then execute the program individually without leaving the web browser. The editor became an effective classroom tool in the flipped learning model utilized in this course. Qualitative data were collected through the use of observations and surveys. The author discusses what was learned about the impact on students' attitudes, learning and quality of work for this class. Challenges are also described, as well as recommendations for enhancements.

## Introduction

The Introduction to Program Design class is required for students majoring in computer systems technology, digital media technology, and electronic & computer engineering technology. Significant changes were made to the Introduction to Program Design class for the fall 2013 semester to better accommodate the diversity of student majors. One of those changes included switching from the Python programming language to Processing (processing.org). The Processing language was originally developed to be a first programming language for those interested in the visual arts. Therefore, it is relatively easy to create visual and interactive programs while still learning the foundational concepts of programming. Processing comes with its own integrated development environment (IDE) that will run on Windows, Linux, and Mac OS X. The Processing IDE is easy to use for beginning programmers. Furthermore, there is a potential tie-in for the electronic students because the IDE and programming syntax in the Wiring (wiring.org.co) and Arduino (Arduino.cc) single-board microcontroller platforms are derived from Processing.

Another change made to the class was a move towards more of a flipped classroom approach.[3] Students were given a series of weekly activities to do on their own time. These activities included online tutorials, readings from online books, and video lectures. Class time was then used to answer questions and work on programming exercises. Two out of three class meetings a week were taught in a computer lab. The class structure and environment were the motivation behind the decision to use a collaborative editor to facilitate cooperation between individual students and with the instructor.

## Collaborative Editor

It was decided to develop a collaborative programming environment that will run within a web browser. Browser-based collaborative programming editors offer several benefits, including the ability to be used globally without installation on the local machine and the ability to be used on multiple operating systems by cooperating programmers.[4,5]

A customized collaborative programming environment was developed using Firepad (www.firepad.io) as the base. Firepad, in turn, is an open source collaborative text editor based on the CodeMirror editor (codemirror.net). The editor is written in JavaScript and will run in most modern browsers. Firepad relies on Firebase (firebase.com) for data synchronization and cloud storage. By adjusting the permissions on the Firebase cloud storage and developing variations of the JavaScript front end, several different modes were created:

- **Student mode** – Any edits by the instructor will be saved to the cloud storage. Students can see the instructor changes, but any student edits will not be saved to cloud storage and will only be seen by the student that made the change. A student can remove their own edits and resynchronize with the instructor by refreshing the browser window.
- **Everyone mode** – Everyone is editing the same workspace. All student edits are saved to cloud storage and are seen by everyone else in this mode. A list of all the connected users is displayed alongside the editor. The border of the editor window is green to indicate that the student edits are saved and therefore seen by others connected to this workspace.

- **Leader mode** – The student in Leader mode can edit their own workspace and changes will be saved to a cloud storage location associated with their user id. This mode allows other students to follow along by starting Lurker mode. A list of all the connected users is displayed alongside the editor. The border of the editor window is green to indicate that the student edits are saved and therefore seen by others connected to this workspace.
- **Lurker mode** – Students can see what other students are doing in their workspaces. A student entering Lurker mode will have to specify which user they want to watch. Once connected to another student it works like the Student mode described above. The lurker can see the leader edits, but any lurker edits will not be saved to cloud storage and will only be seen by the student that made the change. A lurker can remove their edits and resynchronize with the leader by refreshing the browser window. A list of all the connected users is displayed alongside of the editor.
- **Group mode** – This mode allows any student to create a group workspace on the cloud storage. Other students can join the group by simply typing the same group name. All group member edits are saved to cloud storage and are seen by everyone else in this group mode. A list of all the connected group users is displayed alongside the editor. The border of the editor window is green to indicate that the student edits are saved and therefore seen by others connected to this workspace.

When students start the collaborative editor they are asked for a username. To maximize flexibility, ease of use, and ease of administration, the student can make up their own username. A password is not required. This means that students can easily make multiple workspaces by using different usernames, such as, "BobS1", BobS2", and "BobS_greenCircle". Typically the students use their university id as their username. Once students establish a username they can then choose which mode to enter.

The collaborative programming environment is capable of running Processing programs in the browser by utilizing the Processing.js JavaScript library (processingjs.org). The program code within the editor space will execute to an output space alongside or below the editor, depending on the width of the browser window. Anytime a change is made to the code in the editor space the code is re-executed automatically. The execution of the code takes place totally within the browser and does not affect other students. Therefore local changes to the code in the editor will be executed even if the edits are not saved to cloud storage. As a result a student in Lurker mode could make changes to the code in their own view and see the results without affecting the other students. However, edits by a student in Leader mode will be seen and executed by all of the browsers following that leader.

Since the collaborative programming environment is browser-based it is capable of being run on a number of platforms. In addition to running on the Windows-based computers used during class, it has also been tested on MacBooks, Android phones and tablets, iPads, and a Blackberry phone. Small screens and touchscreens make interactions somewhat cumbersome, but they work fine for following along and making small edits.

**Student Perspective**

The students were surveyed at the end of the semester. The survey used a Likert scale where students rated their agreement with a series of statements as 1) Strongly Agree; 2) Agree; 3) Disagree; 4) Strongly Disagree; and 5) Not Applicable. Students also had the opportunity to answer some open-ended questions. Twenty-two students out of the twenty-three in the class responded to the survey. Here are the survey statements and responses:

1. **As compared to just using the classroom project, the editor made it easier to follow along with the instructor when looking at programming examples.**
   Strongly Agree – 16; Agree – 6; Disagree – 0; Strongly Disagree – 0; Not Applicable – 0
   All twenty-two students indicated some form of agreement with this statement.

2. **Being able to see the program run instantly in the browser was helpful in the learning process.**
   Strongly Agree – 13; Agree – 6; Disagree – 2; Strongly Disagree – 0; Not Applicable – 1
   The three students that did not agree with this statement indicated a preference for the way the Processing IDE requires the user to click a button to run the program.

3. **While in student mode, being able to edit and experiment with the instructor's code in real-time without affecting anyone else's view was a helpful feature.**
   Strongly Agree – 14; Agree – 8; Disagree – 0; Strongly Disagree – 0; Not Applicable – 0
   One student did not answer this statement. The others all indicated some form of agreement.

4. **The editor was an effective way for a student to share their code and get feedback during class.**
   Strongly Agree – 14; Agree – 8; Disagree – 0; Strongly Disagree – 0; Not Applicable – 0

5. **The editor was an effective way to share code and get feedback from the instructor outside of class.**
   Strongly Agree – 12; Agree – 7; Disagree – 1; Strongly Disagree – 0; Not Applicable – 2

6. **Group mode was helpful when working with other students on a program.**
   Strongly Agree – 12; Agree – 4; Disagree – 2; Strongly Disagree – 0; Not Applicable – 4
   Some students did not work much with other students which probably explains the four NA answers.

7. **Please describe any benefits of the collaborative editor you found particularly helpful.**
   After aggregating all the students' answers together, features related to all six of the statements listed above were mentioned at least once. In addition, some students mentioned other benefits:
   - Ability to get help by peers without being in the same location.
   - Being able to save code in the cloud storage to access later.
   - Using the cloud storage to transfer code between devices.
   - Ability to run the code on devices, such as phones, that cannot run the Processing IDE.
   - Ability to see the instructor's code from class at home.
   - Ability to work anywhere there is a web browser and an Internet connection.

8. **What improvements or features would you like to see added to the editor?**
   Some of the improvements students suggested are listed below:
   - An option to control whether or not the program will run automatically on each change.
   - Ability to save a program with a file name.
   - Make a sound library available.
   - Built-in examples from a link.
   - An option to display program output in a separate tab or window.
   - Better error messages for debugging code.
   - Ability to see a list of programs that are in the cloud storage.

**Instructor Perspective**

The main objectives of being able to quickly and easily share code in real time were realized. It was very effective to be able to type out and explain code during class while students followed along in the editor's Student mode. In keeping with the flipped classroom ideal of learner centered education, it was especially helpful to pause and let students experiment with the code to see the how their changes affected the output of the program. This seemed to keep students engaged and brought up questions that may not have been asked otherwise. Since their changes didn't affect the original code stored in cloud storage they could then reload their browser window to resynchronize with the instructor's code. The instructor could then continue on with additional modifications to the example code.

Again, in keeping with the flipped classroom model, it was common to assign in-class programming exercises. In most cases students were allowed to work with other students on the exercises. The Group mode of the editor was often used by students to either collaborate in real time on a program or to share code snippets with other students as they each worked on their own program.

Students often used the cloud storage aspect of the collaborative editor to do asynchronous sharing of code with other students and the instructor. For example, a student might copy their program from the Processing IDE to Leader mode or Group mode editor space outside of class time. Then during class the student would do a sort of show-and-tell. This even worked well when the class was held in a regular classroom, in which case the program would be shown on the projector from the instructor's computer. Another example is that some students used the collaborative editor to get help from the instructor outside of class. They would send an email describing their problem and would specify the username or group name they used in the collaborative editor. The instructor could quickly bring up the program and make changes or comments in the code and then email back to the student to review the updated program.

It was anticipated that the collaborative editor would be used by the instructor to give real time help to students while they worked on an assignment at home. Some students mentioned they talked to each other on the phone while editing a program together in group mode. However, that did not happen between the instructor and students. It certainly is a potential benefit, especially if the collaborative editor is used in a distance course.

The Everyone mode of the editor was not used much by students or the instructor. One potential use that will be tested in a future semester is to have a program that any student can add to over the course of the semester. It could be something that the instructor calls up at in the beginning of each class to see what may have changed since the last class period. It will be interesting to see how the program evolves as the students learn new concepts.

## Conclusions

The collaborative editor is not a replacement for the Processing IDE, nor was it meant to be. Nevertheless, it was effective for sharing code both synchronously and asynchronously. It was used heavily by the instructor during class to demonstrate code in real time, which all students found helpful as compared to just following along on the projector. Likewise, all of the students found it an effective way to share code and get feedback during class. In addition, several students liked the ability to edit and run their programs on different mobile platforms using the collaborative editor. From the instructor's point of view the biggest benefit of the collaborative editor was that it helped improve student engagement by encouraging students to learn not only from the instructor, but from each other.

## Bibliography


1. Raines, Claire and Arnsparger, Arleen, Millennials at Work, 2010, http://www.generationsatwork.com/articles_millennials_at_work.php, accessed January 3, 2013.
2. Oblinger, Diana and Oblinger, James (Eds.), *Educating the Net Generation*, Educause, 2005.
3. Bergmann, Jonathan and Sams, Aaron, *Flip Your Classroom: Reach Every Student in Every Class Every Day*, ISTE and ASCD, 2012.
4. Lautmaki, Janne, Nieminen, Antti, Koskinen, Johannes, Aho, Timo, Mikkonen, Tommi, and Englund, Marc, CoRED—Browser based collaborative real-time editor for Java web applications, Proceedings of the 15th ACM Conference on Computer Supported Cooperative Work (CSCW), 2012.
5. Goldman, Max, Little, Greg., and Miller, Robert C., Real-time collaborative coding in a Web IDE, Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST), pp. 155—164, 2011.