Enabling Multi-User Computational Thinking with Collaborative Blocks Programming in MIT App Inventor

Xinyue Deng[1], Evan W. Patton[1*]
[1] Massachusetts Institute of Technology, Cambridge, MA, USA
{dxy0420, ewpatton}@mit.edu

ABSTRACT

Collaboration becomes increasingly important in programming as projects become more complex. With traditional text-based programming languages, programmers typically use a source code management system to manage the code, merge code from multiple editors, and optionally lock files for conflict-free editing. There is a limited corpus of work around collaborative editing of code in visual programming languages such as block-based programming. We propose an extension to MIT App Inventor, a web-based visual platform for building Android applications with blocks, which will enable many programmers to collaborate in real-time on MIT App Inventor projects. We take the position that real-time collaboration within MIT App Inventor will encourage students in a group environment to interact with one another in ways that help them improve each other's understanding and practice of computational thinking practices that may not be achieved in the traditional one user-one project paradigm that is currently provided.

KEYWORDS

Real-time collaboration, App Inventor, visual programming, computational thinking

1. INTRODUCTION

Cloud-based collaborative technologies such as Google Docs have become a central part of how teams work together to collaborate in real time on all manner of content. While real-time collaboration for programming has been explored in research settings, a typical editing pattern in software development involves developers working separately and then merging their changes through a source code management system, such as Subversion or Git. These solutions work well for textual programming languages. However, little work has been done exploring real-time collaborative techniques for visual programming languages, including blocks-based languages including Scratch (Maloney, Resnick, Rusk, Silerman, & Eastmond, 2010) and MIT App Inventor (Wolber, Abelson, Spertus, & Looney, 2011). The remainder of this paper will focus on the challenges and possible benefits of real-time collaboration as they relate specifically to the MIT App Inventor software.

MIT App Inventor is a web-based platform for building mobile phone applications targeting Android. It provides two editors for building an application: a designer where users drag and drop *components*, such as buttons, to lay out the user interface of an application, and the *blocks* editor where program logic is provided using a puzzle block-like language based on Google's Blockly. MIT App Inventor users require a Google account to identify themselves to the service and projects are tied to these accounts. While it is possible to perform group collaboration in MIT App Inventor given its current implementation, this is usually accomplished by student groups creating a shared Google account and trading off control over who is editing using the single account.

We propose a collaborative programming environment within the MIT App Inventor software that will enable multiple users to engage in computational thinking in a real-time collaborative manner. Section 2 describes the related work in computational thinking and collaborative programming. Section 3 illustrates our design and implementation of the collaborative environment. Section 4 presents a discussion that how this system can help users engage in computational thinking.

2. RELATED WORK

Brennan & Resnick (2012) gauge computational thinking with respect to three categories: computational concepts, computational practices, and computational perspectives. They defined "Connecting" as one of the computational perspectives, which involves programming with others and programming for others. By collaboration, programmers are able to accomplish more than what they could have on their own.

With text-based programming languages, programmers usually collaborate with a version control system, such as Git. Guzzi, Bacchelli, Riche, and Van Deursen(2015) presented an improved IDE with support of version control system to help programmers to resolve conflicts and detect problems introduced by others' code. Other than version control system, Goldman, Little, & Miller (2011) demonstrated a real-time collaborative web-based editor for the Java programming language.

Collaboration in blocks-based programming languages has typically been done via remixing, such as in the Scratch language (Maloney et al., 2010) and MIT App Inventor (Wolber et al., 2011). In remixing, a developer publishes an application publicly and others use it as a starting point for a new application. This remixing behavior makes iterate development between two developers more difficult because the project, rather than some subset, is the basis for remixing.

Greenberg & Gutwin (2016) highlight key challenges in enabling awareness in collaborative environments. We leverage their findings by codifying awareness information via the locking mechanisms proposed in Section 3. These locking mechanisms allow users to direct awareness of their peers by synchronizing access to the

environment on a per-user basis. Gross (2013) provides a more in-depth review of awareness research.

3. DESIGN AND IMPLEMENTATION

Our collaboration system is mainly designed for group course projects of 2-4 students in middle school, high school, or college. The system will satisfy the following features:

1. Users are identified by their email address and share projects with others by email address. The user who creates the project can change others' access level of the project. The access level includes read, in which users can only view the project, and write, in which users can both view and edit the project.
2. Users can know who is currently working on the project, and the components or blocks that each individual is currently working on.
3. User can see others' changes simultaneously. There are several cases in MIT App Inventor:
   a. When users work on different screens, their changes will not be shown until switching screens.
   b. When users work on the same screen, and they work on the same editor, they can see the others' change immediately on the editor.
   c. When two users work on the same screen, and one works on the designer editor, and the other works on the blocks editor, the one on the blocks editor can see new blocks when the one on the designer editor adds a new component. When the one on the designer editor removes a component, the other will see blocks related to that component disappear.

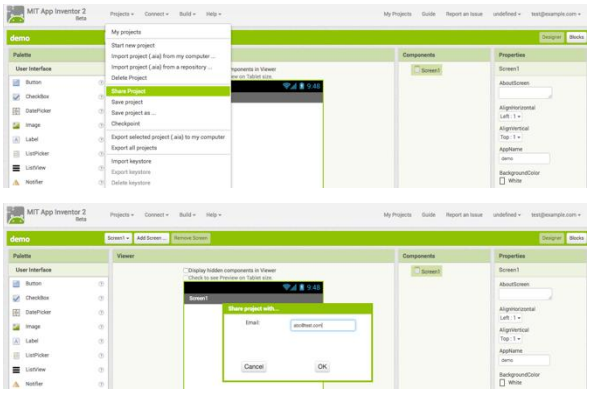

Figure 1. Share project by entering user's email address

3.1. User Interface Design

A user can share a project with others by providing their email address. Figure 1 shows the user interface of sharing a project. Once the project is shared successfully, the other user can find the project in her project explorer. Users can know who has opened the project by the colored square in the project title bar. When user hovers on the square, it will show the user's email address. The color of the square indicates the user's color. It is used to identify which part of the program a user is editing.

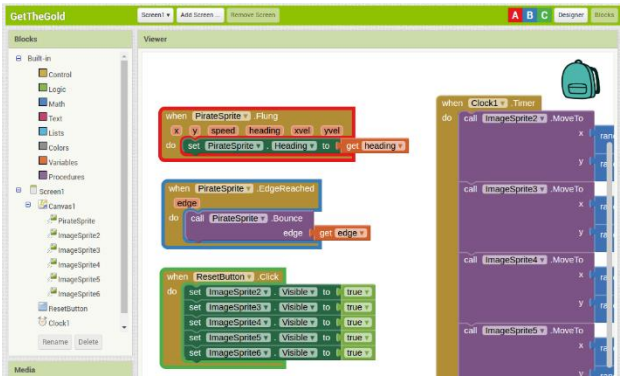

Figure 2. An example of collaborative block-based programming in MIT App Inventor. This project is shared within four users. The user can see the other three users, A, B and C, on the project title bar. The block that each user is editing is highlighted with the user's color.

3.2. Collaboration Server

In order to show others' changes immediately, we use publish-subscribe pattern to send updates from one user to others. Publish-subscribe pattern is a messaging pattern, where senders can send messages to a channel, and receivers who subscribe to that channel can receive the messages. We decided to build a NodeJS server for web clients to communicate about collaboration, which runs separately from the MIT App Inventor server, so it is easy to be managed. MIT App Inventor clients connect the collaboration server with sockets. We use Redis, an open source library for in-memory data structure store and publish-subscribe pattern, to publish and subscribe updates (Redis Contributors 2017), and all messages will be in JSON format. The client will translate changes into JSON documents and send them to the collaboration server over a specified channel. The server will apply operational transformations on JSON documents to make sure changes are published consistently to all subscribed clients. Then, clients translate JSON document into events that update the code and run the events on their individual systems. Therefore, the copies of the code of all clients will eventually be identical.

3.3. Channels

Each MIT App Inventor client is both publisher and subscriber in the system. Clients will subscribe to three kinds of channels:

1. User channel: The user channel is specified by the user email address. Each client subscribes to only one user channel. When users share a project, they publish the project and user information to others' user channel. Therefore, other users will be notified that a user shares a project with them, and that project will appear in their project list.
2. Project channel: Project channel is specified by project id. (Each MIT App Inventor project has an id that is unique to the MIT App Inventor server.) When a collaborator opens a project, he will subscribe to that project channel. This channel is used for project-level messages, such as when other collaborators open or close the project, or when components are added, modified or removed. When a collaborator publishes

changes to the project channel, all active collaborators on that project will be notified.

3. Screen channel: The screen channel is specified as combination of the project id and the screen name. This channel is used to publish changes about blocks. Each screen has its set of blocks. Users subscribe to this channel when they open the block editor of a screen. After subscribing this channel, all changes related to blocks in this screen will be published to the channel.

4. DISCUSSION

The collaborative programming environment within MIT App Inventor provides users a new approach to teach and learn. For example, it enables "teacher-student" or "mentor-mentee" roles inside MIT App Inventor. Teachers can share the projects with students in read-only mode to demonstrate ideas and demos. Students can work on group projects after school, because they can collaborate remotely. As MIT App Inventor is built for students and novice programmers, the collaborative programming environment gives them an opportunity to develop their teamwork skill at an early stage. Also, while developing applications collaboratively, users can learn how to resolve conflicts.

This new collaboration mechanism for MIT App Inventor touches on all four of the key computational thinking practices of Brennan and Resnick (2012). Multiple users can incrementally and iteratively build small units either in isolation or together depending on the complexity of the tasks and expertise of the individuals. Users can explore different debugging techniques to assist one another in addressing problems in the code. Reuse and remix of code can happen on a much finer time granularity on the order of seconds or minutes. Lastly, users can work together to help one another understand and exploit abstraction and modularization techniques within a program.

One challenge for collaborating with visual programming language is that it is hard to understand others' thought process. With the text-based programming language, programmers can know others' plan via comments. However, it is hard to place comments in visual programming environment without disrupting actual programming logic. One way we can handle it is to add a screen for comments, so users can toggle the comments screen as they need. Another way to help users to understand others is adding a communication channel, so that users can exchange their ideas while they are programming.

Our technical approach is not restricted to MIT App Inventor, as it builds on Google's Blockly. It can therefore be applied to other visual programming languages, such as Scratch. It is easy to integrate socket and publish-subscribe pattern into the system.

5. CONCLUSIONS

We presented a collaborative programming environment within the MIT App Inventor software and provided technical details of an implementation of real-time collaboration. In future work, we will evaluate the effectiveness of the collaboration with novice and expert users of MIT App Inventor to better understand how students use the system to collaborate.

6. REFERENCES

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).

Goldman, M., Little, G., & Miller, R. C. (2011, October). Real-time collaborative coding in a web IDE. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 155-164). ACM.

Greenberg, S., & Gutwin, C. (2016). Implications of we-awareness to the design of distributed groupware tools. *Computer Supported Cooperative Work (CSCW), 25*(4-5), 279-293.

Gross, T. (2013). Supporting effortless coordination: 25 years of awareness research. *Computer Supported Cooperative Work (CSCW), 22*(4-6), 425-474.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16.

Redis Contributors (2017). *Redis Publish-Subscribe message pattern*. Retrieved February 4, 2017 from https://redis.io/topics/pubsub.

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). *App Inventor*. O'Reilly Media, Inc.

Guzzi, A., Bacchelli, A., Riche, Y., and Van Deursen, A. (2015). Supporting Developers' Coordination in the IDE. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*