Middle School Experience with Visual Programming Environments

Barbara Walters and Vicki Jones
vanbara, Inc
Apex, NC USA
Email: {barbara.walters, vicki.jones}@vanbara.com

***Abstract*—Middle school students often avoid programming endeavors, thinking they are boring and non-productive. We have been successful engaging and maintaining interest in programming through the use of App Inventor 2 and a curriculum of our design. This paper details the features in App Inventor 2's visual programming environment that we believe are significant contributors for engaging students and facilitating their understanding of abstract concepts.**

Keywords—visual programming; middle school; teaching programming;

I. INTRODUCTION

As computing professionals who love programming, we have fully embraced visual programming environments to entice the next generation of programmers and technologists. As volunteers at local schools, we have developed curriculum and activities to teach programming using App Inventor 2. In this paper we discussour experience and major contributors to our success.

We have engaged in outreach activities to attract students to engineering for most of our professional careers. More recently we have tutored middle school students in math and science. We found their lack of exposure to programming troubling and decided to become more proactive.

In the 2014-2015 school year we started using App Inventor 2. We worked with nine groups of students, in year-long weekly after-school clubs, as enrichment in an existing class and as week long summer camps. There were over 150 students involved in these activities.

Teaching middle school students to program is wholly dependent on keeping them interested long enough to succeed. We have discovered that middle school students are interested and remain engaged in programming activities that produce an outcome they can produce with ease, creatively customize, and share with pride. Our program allows them to do that using the App Inventor 2 visual programming environment to develop customizable apps.

Teaching hundreds of middle school student to program is wholly dependent on enabling their teachers to succeed. App Inventor 2 is easy to use by anyone, regardless of their experience with technology. Our experience in North Carolina is that little is required in the way of infrastructure not already available in many school and community locations. App Inventor 2 and our curriculum provide ample opportunity for enriching the middle school curriculum.

II. VISUAL PROGRAMMING IS REQUIRED

Our goal is to teach middle school students how to program. These students are mature enough to follow multi-step instructions, understand math concepts such as variables and the coordinate plane, and can read and understand words such as "orientation." In order to engage students, we found a visual programming environment was a requirement. Many of our students have been previously exposed to programming activities that left them with a negative impression of programming. However, because we offer to teach students how to make apps, we have no difficulty attracting students to our program and overcoming their initial hesitation. Producing apps is the hook for engaging our students. The ease of the visual programming environment keeps them, and their teachers, coming back.

Many of our students have parent(s) who program professionally for companies like Cisco, SAS Institute, Red Hat, Epic Games and others. The parents, including the authors, love programming and tried to share the experience with their children. They used a variety of text based languages such as Python, Java and Javascript. In general, these attempts were unsuccessful because the students simply did not like using these languages. They disliked the programming environments, they were impatient with the syntax requirements and they struggled to create something they wanted to show off.

Searching for more engaging programming environments, we found Scratch. In the fall of 2013 we began using Scratch at several middle schools as part of a year-long enrichment program. Acquiring the necessary computer facilities and internet access were no problem so this made Scratch very attractive. Since we did not find many materials to use with the class, we created activities that are very similar to those included in Google's CS First.

Through that experience we discovered that one very important factor in motivating students is the ability to customize a program and make it their own. We had difficulty supporting those desires with Scratch. After students completed several Scratch activities and understood the basics

978-1-4673-8367-7/15/$31.00 ©2015 IEEE

of programming, we let them choose one of three types of projects: a program that demonstrated a science concept, told a story or related a set of facts on some topic. Initially the students were excited, but for many of them the enthusiasm faded. Rather than programming, we spent many hours searching and manipulating images for their projects.

It was our experience that the functionality provided by Scratch was not sufficient to engage our target age group and encourage them to continue programming.

III. Our Experience with App Inventor 2

After using Scratch, we looked for alternatives. We experimented with Snap! and Alice before finding App Inventor 2. App Inventor 2 provides a rich programming environment with a variety of components: traditional UI components (button and label), animation components (canvas and sprite), sensors (accelerometer and location), social media, storage, connectivity and so on. From the beginning, our students recognized that they were not restricted by the environment. Importantly, to them this was not merely a vehicle for teaching programming, but an environment where they could create fully functional apps.

The schools in North Carolina are well equipped with computers and wireless networks. Bring Your Own Device programs are becoming common. However, App Inventor produces Android apps and we found no schools had those devices; Apple devices were common. Before our program could be deployed, we helped teachers obtain devices through programs like Donors Choose and donations from corporations and individuals.

App Inventor provides an emulator. There were numerous issues with the emulator, from long start up time, lack of functionality (you can not shake an emulator) and frequent updates. We chose not to use the emulator and instead used the AI2 Companion app. This also requires frequent updates but was a better choice for our program.

From the very beginning of our program, students have the opportunity to customize their work. We were struck by how important customization has been to our students: we attribute much of our more recent success keeping students engaged to providing opportunities for them to exercise their creativity and make their product unique.

Initially our students were eager to make their own apps (not ours), but they have no idea what skills they need to be successful. Our approach is to have students create 5 apps of our design, then guide them in creating an app of their design. By requiring them to make our apps, we can build skills and introduce new concepts gradually. Because App Inventor 2 provides such a rich set of components, it was easy for us to create apps that have a variety of functionality and afforded the opportunity to gradually introduce computer science and math concepts. The depth and richness of App Inventor 2 is very beneficial in keeping our student's interest as well as convincing them that this is a "real" programming environment.

Each of our five apps teaches new concepts and introduces new components. They include several games, apps that perform text manipulation used in Mad Libs or Hang Man and utility apps that use API's of other apps. The apps access data that may be stored in the app, on the device or on the cloud. Some apps use social media. In addition, each app allows the student to exercise their creativity by customizing the app. We had very few students that were satisfied by merely producing our app. Initially, due to having minimal skills, the students modified the appearance of the app. This led to lively conversations about intellectual property rights and copyrights.

As they gained skills they began to change the behavior of the apps as well. They added or modified blocks to change the logic of a program. They added components such as social media and storage components. They also started thinking in more concrete terms about the apps they would like to make. They started thinking about what was available in App Inventor 2 and how they wanted their apps to look and behave.

We attribute our success using App Inventor 2 to engage our students to several factors such as creating something concrete and shareable and facilitating creativity. Another contributor to this success is the App Inventor 2 user interface. For the students it is important that it is easy to use. As educators, we appreciate the support it provides in reinforcing computer science concepts. We volunteered with several teachers who teach programming in high school and they were surprised and pleased at how quickly students grasped the basics of computer science using App Inventor 2. The teachers commented on how difficult it was to teach some of these concepts using their traditional methodology and with this environment, much younger students understood and applied important computer science concepts.

IV. Usability and App Inventor 2

In our first app, we explain fairly complex concepts regarding user interface, components, hierarchies, property sheets, event driven programming and methods. Our students have no problem grasping these concepts because App Inventor's user interface supports these concepts.

In the design view, students are introduced to the concepts of user interface design, components and properties. Once a component is added to the project, it can be selected and its property sheet made visible. At a glance, the students see what properties they can control. In follow-on apps, we discuss application life-cycle and initial conditions. The property sheets are ideal for this discussion.

In the design view they are also introduced to hierarchies based on the way components are organized. A screen is a parent to a canvas and a canvas is a parent to a sprite. Setting the height of a component to the value "Fill Parent" makes this relationship easy to understand.

When they switch to the blocks view and select a component, students easily identify the getters and setters because the names (usually) match the names in the property sheet. Getters and setters are similar colors and appear distinct from other methods.

The blocks view makes it is easy to understand the behavior of a component. When selected, each component displays the events it can detect as well as its methods. Together, the properties, methods and event handlers define the behavior of the component.

Because methods are organized with a component, it is easy to explain that methods are named programs associated with a type of component. Later, when students create procedures, they are already familiar with the concept of a named program. We explain methods are named programs associated with components and procedures are named programs that are part of an application.

Event-based programming is quite natural as well. We start by having students create apps that respond to user events: a finger dragged across the screen, shaking the device, pushing a button. Because they understand user initiated events, it becomes easier to understand other types of events that were not caused by a user action or a sensor. Discussing application life-cycle in terms of events (Create, Start, Resume, Pause, Stop, Destroy) is easily understood, even though all of these events are not surfaced in App Inventor 2.

The operators in App Inventor 2 make it very easy to explain data types. Unlike programming environments where data type is expressed by a shape, in App Inventor 2 data types are color coded and match the color of their operators. In the blocks view, creating a value of a particular type and the operations performed on that type are grouped together, subtly establishing important relationships in the mind of the novice programmer. Many operators will not allow a connection to a data block if it is not the correct type. This re-enforces the concept of data type and provides a natural opportunity to discuss computer science concepts regarding data. Some operators allow connections to any data type. This provides an opportunity to discuss data conversion. Many data types can be converted from one type to another, such as when a number is converted to a string so it can be displayed. This is why the text operators accept most data types. But the conversion is hidden which can be a source of confusion if not explained to the student.

Most property "getter" and "setter" methods require a specific data type, for example true or false for the property "visible". These methods do not require a data block of the required type. We have encountered a few examples of students connecting an incorrect type, but this has been relatively rare.

In addition to being engaging for middle school students, the richness of supported activities in App Inventor 2 allows additional educational advantages. For example, having colors as a data type and blocks to create new colors leads naturally to discussions on binary and hexadecimal representation. Throughout our curriculum we take advantage of other opportunities for enhancing traditional middle school subjects.

Overall we are delighted with App Inventor 2 and are reluctant to criticize because, as an open source project, we could contribute code and we have not prioritized the time to do so. The following are ideas for improvement. There is only a single area for programming blocks and organizing blocks for anything other than a trivial program is difficult. There is no way to create a library of reusable components. Image Sprites have only a single property for heading which is used both for rotating an image as well as setting the direction for movement. Independent control of these is often desirable. The most common mistake our students make is in typing the name of a media file. If there were blocks that contained the list of media files, this type of error would occur infrequently.

We personally invested hundreds of hours learning how to best use App Inventor 2 to meet our goal of teaching fairly young students how to program. There are lots of resources available on-line that are now used by our students after they completed our program. What we did not find and had to create ourselves, was an age-appropriate introduction to programming that included the computer science concepts we believe are essential.

V. App Inventor 2 Makes Math Real and Relevant

App Inventor 2 makes it easy to demonstrate math concepts in a very concrete way. Many Common Core Math concepts are reinforced in our activities. Teachers are excited that the students see the relevance of these concepts as they build apps.

For apps that use the Canvas component with either image sprites or displaying text, the student becomes accustomed to using the graphics coordinate system to position objects. They have to resize images and scale them so they maintain the correct appearance, using the ratio of the screen size and image size.

Transformations such as rotation, translation, dilation and reflection of image sprites is used in one of the gaming apps and we introduce and define those terms. Variables are used to maintain state and in calculations.

Time units are used for moving image sprites as well as clock events. Order of magnitude is used to describe human time (seconds) compared to the time units used in App Inventor 2 (milliseconds).

Boolean values and operators are used regularly in control expressions and are used to implement the rules of a game.

Probability and random numbers are used in one of the game apps and are common in student-designed apps.

Bits, bytes, binary and hexadecimal representation are easily explained through the construction of colors.

These are just a few of the math concepts used in our activities. Our students are familiar with some of the concepts, such as the coordinate system, but some are not introduced until 8th grade or later. Our students are excited to use their math skills and enjoy understanding how to use them in making apps.

VI. Student Ownership

For each of our apps, the student builds the entire app: the user interface, the program logic and finally customization. In our initial pilot we miscalculated the complexity of one of the apps and it took our students nearly eight hours to complete it. When we realized how long it would take, we offered a

partially completed project to shorten the time to completion. Our students rejected the offer, preferring to complete the app on their own, at their own pace.

In another setting, we worked with teachers who taught programming electives and asked if they had suggestions for improving our program. They suggested that we create App Inventor 2 projects that are partially completed and have the students add only the parts that are related to the concepts being taught. This reduces the amount of time the students spends on a particular app.

Through post-project surveys we found students did not like the partial project approach. In all cases where we offered a partially completed app, the students responded with much lower ratings compared to projects built entirely by the student. After a few attempts with partial projects, we discontinued this approach.

We do not know why partial projects are not as well accepted and think this is an area that may merit further study. One hypothesis is that the process of producing a correct, well-behaved and attractive program is similar to the process of creating a well-constructed piece of literature. Both engage a student's creativity, both have a set of well-established best practices and require skills attained through practice in order to produce a high-quality end-product.

The act of writing is an iterative process, just like the programming process. Rarely are students asked to contribute just a portion of written work to an existing document. Instead, they learn to write by making many attempts at creating documents with appropriate beginning, middle and end, refining their skills and their expertise over time. The premise is similar. Partial programming is equally unsatisfying as editing or completing literary works.

VII. Measures of Student Success

While we provide pre- and post-course assessments as well as quizzes for each project, these have been used only in classroom settings where our content is used for enrichment, not in the summer camp or school club setting. Our primary measure of success is whether, after creating our 5 apps, a student has learned enough that they can design and implement an app by themselves.

Each of our apps starts with a specification describing the appearance and behavior of the app. This is followed by a description of the App Inventor 2 components that are used. For this last project, the student is given only a description of the behavior of the app. In most cases we have asked them to make a magic eight-ball app. Many of our students are not familiar with the magic eight-ball, so we demonstrate how it works and give them time to play with it and discuss it.

The students then create their design which includes both the user interface and the behavior of the app. They choose the components that support the desired behavior, create the associated programs and complete the app.

This project gives them the opportunity to explore and use components that have not been used in our apps. Our students are able to complete an app of their design and we have been delighted at the variety of implementation techniques and the creativity they demonstrated. They proved they had learned a great deal and that they could expand their own knowledge independently.

We expect our second year students to continue making a few apps following our instruction in order to introduce additional concepts and teach more advanced techniques. In addition, we look forward to them exploring their own ideas for apps, making trade-offs in functionality versus complexity and increasing their expertise.

VIII. Next Steps

Our activities have been used in for a one-week summer camp, as enrichment included in another course or for a year-long after school club. For the second year, our students will continue to use App Inventor 2 to create apps of our design and implementation. In addition, the experienced students will mentor novice students as well as creating apps of their own design. We will introduce robotic programming as well as data collection through sensors.

But after the second year, we believe it will be appropriate to introduce more traditional languages and development environments. We are also concerned that at this stage, expertise within the classroom will be essential and there are not very many teachers knowledgeable in this area.

One language we are considering is Python. There are a wide variety of frameworks that work with Python, from gaming to data science. Whatever language is chosen as a follow-on, it must provide a functionally rich environment and the end product has to be as exciting and fulfilling as the apps produced by App Inventor 2.

IX. Acceptance in Schools and Teacher Engagement

Our activities were offered in a variety of environments: after-school club, in-class enrichment, summer camp. We believe this approach will also be successful as an elective course but would require acceptance at the district or state level and additional educational material. Instead, we plan to continue to offer the activities as enrichment or in informal settings.

Most schools have the infrastructure to support App Inventor 2: appropriate computer systems and browsers; and most schools allowed installation of the App Inventor emulator. Most of the schools provide Google accounts for their students. None of the schools had Android devices available; all had IPads. The private schools purchased Android devices so they could use our activities. We donated a set of Android tablets to one school and several others used Donor's Choose or 21st Century grants to pay for the devices.

A professional teacher was always involved, either in leading the activity or in partnership with us; we worked with 10 different teachers. They were certified in a variety of fields: elementary education, gifted education, math, science, Project

Lead the Way middle school programs, media specialists and computer programming specialists. It surprised us how reluctant some teachers were to learning the material. While all of them were supportive and very pleased with the how well the students were doing, only half of them tried the activities themselves.

In the classrooms where we were not involved and the teacher did not master the material, the students were still successful. Our activities are fairly prescriptive, with test and debug steps incorporated as the app is developed. This limits the scope of problems and we have encouraged the students to ask each other for help. That strategy has worked so well, that in the classrooms where we are present, we are rarely involved while the students are actively programming. We found ourselves much more in the role of coach and as a facilitator during discussions.

We believe in order to offer the opportunity to learn programming to a wider group, engaging teachers is essential. Teacher mastery of the content is not necessary for student success, but teacher buy-in is essential. The activities will not be offered unless there is a teacher willing to sponsor the activity. Anecdotally, teachers who plan to teach the material at a summer camp, in a community center or in a fee-based after school program were interested in learning the material themselves. We believe this is because they will be reimbursed for their efforts.

To encourage teachers to learn the material, we have engaged several of them to teach at community centers. We are also setting up an online course and community for teachers. The course will be offered during specific dates, not as a do-it-yourself program, with assignments and assessments. When a teacher completes the program, we provide a certificate of completion and will support them in teaching the activities in informal settings. Our hope is that once these teachers have mastered the material, these teachers will use the activities in the classroom as well.

There will continue to be an issue with the lack of Android devices, but we are also providing guidance regarding grants that will help purchase the devices and we have been astounded by the generosity of people donating via Donors Choose, especially people who are outside the school district and seem to donate based on the prospect of encouraging students to code.

X. Conclusions

In the 2014-2015 school year our activities were used in five schools and in two summer camps. Over 150 students, mostly fifth through seventh graders and about a dozen high school students participated. For 2015-2016 we are expanding where we teach and will offer the program in local community centers. Based on recognized need, we also plan to develop on-line instruction and a community for teachers.

Why are our students asking for more? Encouraging their friends to participate? We believe it is because they have made something unique, that they can show off with early and continued success made possible by App Inventor 2. We hope to share the enthusiasm by reaching more teachers, enticing them through a curriculum with obvious value for enhancing the curriculum and an environment that is not intimidating.

We are interested in pursuing research to vet the hypotheses presented here: Are students more interested in programming when creating apps, not just applications? Do students value creating their own product in its entirety and customized to their liking? Do students better understand the math concepts they have used in programming? How do we assist teachers in leading programming/computer science discussions?