A Tablet-Based Lego Mindstorms Programming Environment for Children

Stephanie Ludi(✉)

Department of Software Engineering,
Rochester Institute of Technology, Rochester, USA
salvse@rit.edu

Abstract. Tablets, such as the iPad and Kindle, provide a portable platform for children of all ages to explore various content through apps and interactive books. The use of gestures provides a means of interaction that is intuitive to children as a means of navigating apps or activating media-based content. The tablet as a programming platform is unique in that the gesture-based skills used in other apps are extrapolated and applied to computational thinking skills and interaction with a robot, which maneuvers based on the child's creation. This paper describes the workflow and user interface design to facilitate Lego Mindstorms NXT programming by children.

Keywords: Children · Mobile · Programming · User interface · Programming

1 Introduction

First notebook computers where viewed as a portable technology for children to use, then tablets were introduced. The advantages of tablets over notebook computers are the price, weight, and ability to use touch and gestures as the primary input mechanism over the keyboard and mouse. Touching an item as a means of the exploration of physical objects. Translating that into taps, pinches, swipes and other gestures has enabled children to explore content on the tablet.

Creating content is also another avenue of interaction. Usually artistic creations are associated with the type of content that children create, but content can also include songs, video, and even interactive content. To extent the type of content and the means of interaction, the pair of tablet-based project described in this paper are those of robotics programs that run on Lego Mindstorms NXT robots. The software line is called JBrick. JBrick for iOS enables children to create their own Lego Mindstorms NXT program by dragging and stacking blocks of programming elements or structures, an example of visual programming (as opposed to typing code).

Visual programming is a popular means of teaching programming to children. Examples include MIT's Scratch, Microsoft Kodu, and Lego Mindstorms' own programming language, NXT-G. The use of gestures, the tablet platform, and the extended metaphor of blocks for portraying program structure in Lego Mindstorms NXT programs are the focus of the JBrick for iOS project.

The use of robotics such as Lego Mindstorms as a means of engaging children in STEM (Science, Technology, Engineering, and Math) has been the subject of several

P. Zaphiris and A. Ioannou (Eds.): LCT 2015, LNCS 9192, pp. 106–114, 2015.
DOI: 10.1007/978-3-319-20609-7_11

classroom and outreach projects [1–3, 5]. For this work, the means of interaction using a gesture-based, tiled approach is the main focus. Cartoid has been developed to enable children to write simple programs and control devices such as Lego Minsdstorms robots, but the program construction, based on Scratch, uses simple taps to select commands that are stacked in a linear manner [7]. A tangible approach, as in Robo-Blocks [4], requires physical blocks to construct a program as well as debugging the program. The authors noted issues with some student's attention, especially with young children.

The notion of blocks as a means of building programs is also used in the Lego Mindstorms NXT-G environment. In the case of NXT-G, programs are built by connecting blocks along a horizontal line, as shown in Fig. 1.

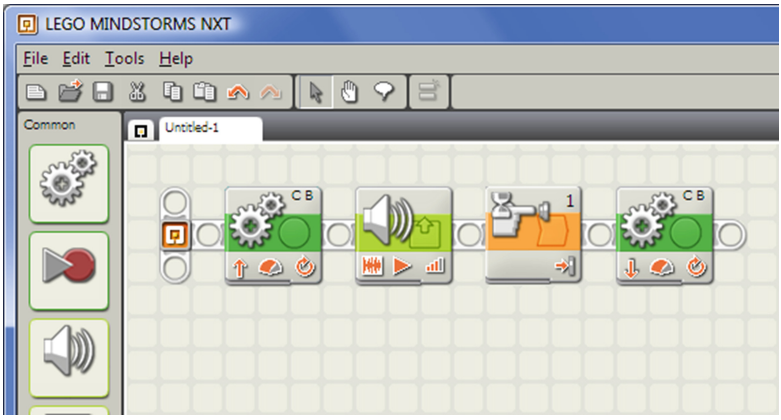

Fig. 1. Screenshot of Lego's NXT-G software (cropped)

The program, as in the case of other Lego Mindstorms programming environments, is controlled with the keyboard and mouse. In the case of NXT-G, attributes are set in a pane at the bottom of the window. The main window is often full screen, with the attribute pane being at a significant visual distance from the program. JBrick takes a vertical program building path, with attributes being visually near due to the smaller size of the screen. The vertical approach is similar to MIT's Scratch, as shown in Fig. 2.

Like NXT-G, Scratch is traditionally mouse-driven. However Scratch is used to create animations and simple games. JBrick for iOS takes a different direction with block design, as well as selection since the use of such narrow blocks in a tablet app would be difficult for a young child to accurately select. More detail will be presented in Sect. 3.

2 User and Task Analysis

2.1 User Profile

Lego Mindstorms are popular in many classrooms. However, many classrooms have a limited number of computers for children to use when programming. The smaller, portable, and low-cost tablets can be a more affordable solution.

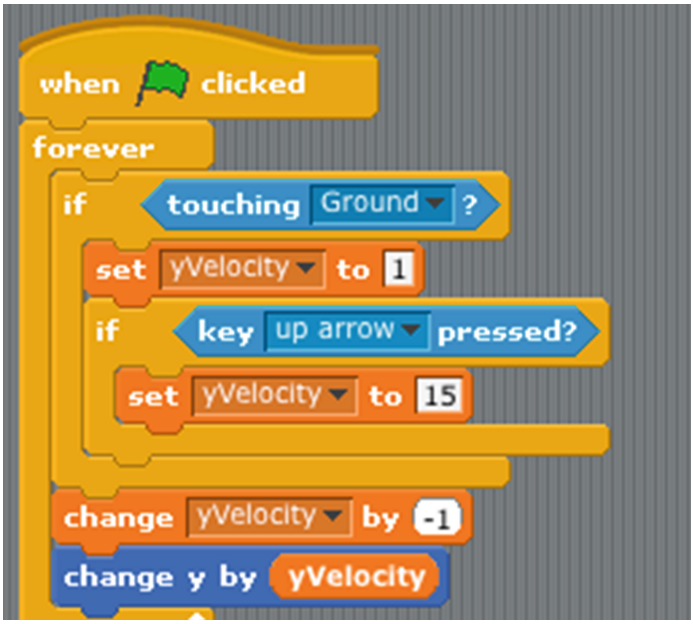

Fig. 2. Screenshot of scratch program

Students, specifically children aged 7 and up, are the primary users of JBrick for iOS. They will be the focus of analysis for this paper. Educators and parents are also stakeholders as they will set up, and in the case of the classroom environment they will oversee the use of several tablets and robots.

Based on observations from past robotics programming camp and FIRST Lego League team participants, the user profile for a child user consists of:

- Age: 7 and up
- Education level: 2nd grade (US) or above, where the child can read and comprehend 5-6 word sentences; also at minimum basic addition and subtraction
- Visual Acuity: with correction 20/20 though up to 20/200 without correction
- Motor skills: Use of a finger or stylus (if motor impaired) for taps, swipes, etc. 2-handed typing not needed
- Programming expertise: None required

Previous experience with 9-17-year old kids [8] has shown that students who are less familiar or confidant with the keyboard have more issues creating and working with their programs. Removing the keyboard and mouse hardware from the equation, as well as enabling the student to control placement of the device when programming offers more flexibility for physical and environmental differences.

The threshold for skill level is set low, as the programs created can be as simple or complex as the student can design. The user interface itself was designed to work with both younger and older students, with varying hand/finger sizes. These attributes facilitate system usability.

2.2 Programming Workflow

The goal is to make the JBrick and Lego Mindstorms NXT programming portable for kids. The previous, desktop version of JBrick as well as other robotics programming software runs on traditional desktop computers or laptops. The child's workspace and computing environment revolves around where the computer is placed.

The desktop version of JBrick's programming workflow was the basis for the tablet workflow in terms of the types of tasks to be completed (not the means of completing these tasks). The foundational programming workflow is shown in Fig. 3.

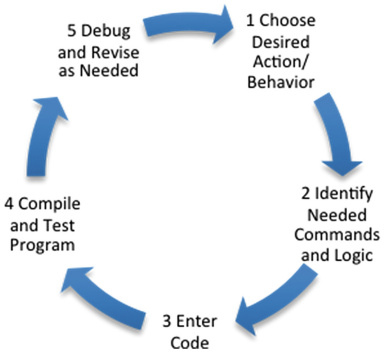

Fig. 3. Programming workflow using JBrick

Programming involves more than entering in code, but it includes the decision as to what features or robot behaviors are desired/needed. Next, design is conducted in terms of thinking algorithmically takes place. Then the process of mapping these requirements to the language via the development environment takes over. The tasks of creating a program, compiling the program, downloading the program to the robot, running (testing) the program, then repeating the cycle as needed until the result is desired is typical regardless of which programming environment used. The differences come out in the user interface design, in order to meet user needs and support the workflow.

3 System Design

Due some external constraints, the system design has a client-server architecture. The NXC (Not eXactly C) compiler that is used for JBrick runs on the PC. As such, the compilation must be accomplished on a PC rather than the iOS device. As shown in Fig. 4, the iPad sends the NXC program to the server. The server then compiles the program and sends it to the Lego Mindstorms robot directly via Bluetooth.

In some respects, this technical limitation would be negative, but in using a Web Service approach, many iPads can be managed from a single computer. As a web service, the PC does not have to be in the same room (a feature to be developed in the future).

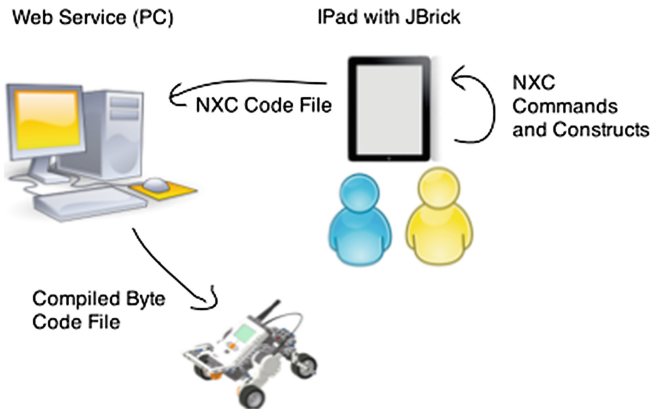

Fig. 4. Overview of JBrick for iOS

4 User Interface Design

The primary tasks associated with the user interface are those related to creating and revising a program. However the entire workflow will be presented.

Take note that the blocks have minimal text, in order to minimize cognitive and visual overload. The size of the blocks is such that small fingers or adult sized finger can interact with blocks easily. Volunteers with different hand sizes provided some initial feedback on block interaction. Since the fingertip is a small size, the outline or anywhere within the block is recognized and triggers block selection. The use of bright colors and icons helps reinforce the type of blocks (and thus concepts used to program the robot).

4.1 Creating and Editing a Program

JBrick for iOS runs on an iPad or iPad Mini as a meaningful amount of screen size is desired as the program workspace. Single finger gestures are used and the number of gestures is minimized in order to aid in ease of learning and memorability. As shown in Fig. 5, programs are built using blocks. The blocks represent constructs, variables, sensors, motors, etc. Sequential commands are laid out top to bottom in a vertical line, while constructs such as if's and repeats have blocks that are stacked within them to indicate nesting and sequence. Each type of block has an associated symbol and color to indicate the type (to help with readability and scanning the program).

Blocks are added to the program by selecting the type of block desired form the menu at the left. Selection is in the form of a long tap, where the feedback is the appearance that the block is lifted off of the screen (or floating on top of it). Upon selection, the user drags the block to the desired position. Blocks that become the body of a construct are offset in order to be distinguishable and to also to effectively use the horizontal workspace.

When a block needs to be set in between two existing blocks, the child simply places the block in between the blocks and releases their finger. After placement in the

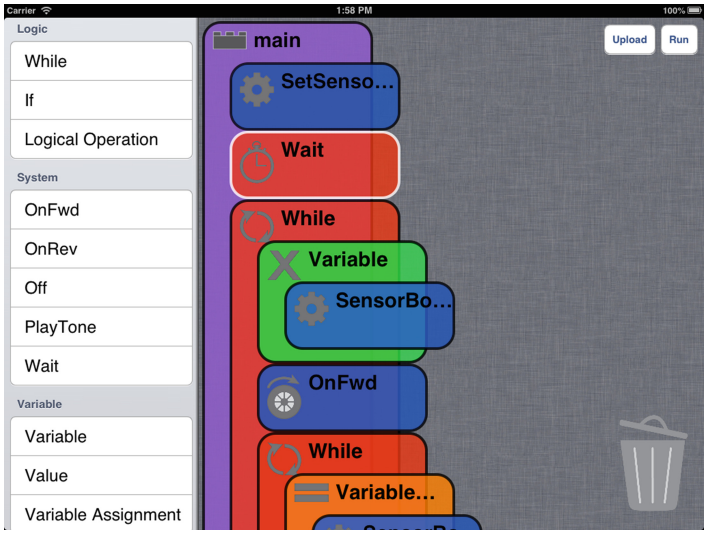

Fig. 5. Overview of JBrick for iOS user interface

program, blocks can be moved by selecting them with a long tap (which picks up the block visually) and then placing it where desired.

Deleting a block in a program occurs when a block is moved over the trashcan icon in the far side of the screen. If the child tries to place a block in a location that is not allowed such as out of the program workflow, the block will snap back to the menu pane and audio feedback is provided.

The child sets any attributes (such as which motor to use or how many seconds to move forward) by tapping the block and then selecting (and as needed entering the value) for the desired attribute. The attribute menu, shown in Fig. 6, is reveals on the right side of the screen only when a block is selected.

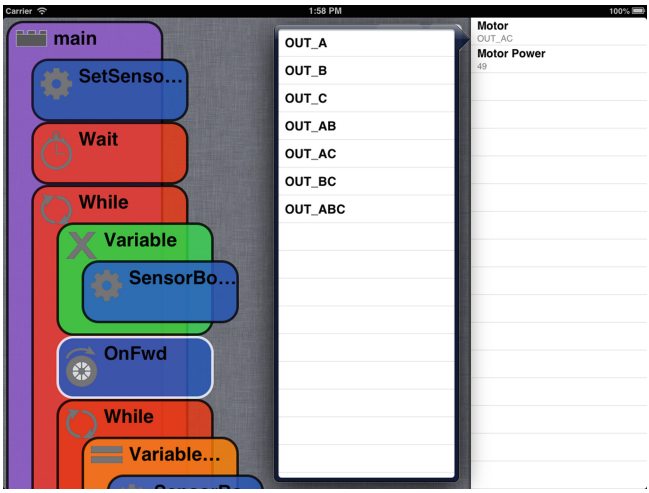

Fig. 6. Overview of JBrick for iOS screen with attribute menu displayed

As the program grows in length, the program flows down and the child scrolls the screen up or down to navigate the length of the program.

The Attribute pane and the command pane slide in and out when the task demands their presence, but the child can also swipe them in or out as desired. As needed a secondary pane will slide out if the child needs to select an item form a list such as the port for a motor. The labels used (e.g. Motor A) correspond to the labels used on the robot itself.

The attributes and command names map to those used in the NXC language. For the attributes, the use of A, B, and C are associated with the ported on the Lego Mindstorms robot. The command names such as PlayTone (to play a sound) were kept to enable students to then transition to the text-based NXC programming language. Currently programs created on the tablet can be opened in the desktop versions of JBrick or BricxCC.

4.2 Compilation and Downloading

The programs that the children create appear graphically, with the complexity of the NXC code managed in JBrick. The programs created can always be compiled, so the child can focus on computational thinking skills and problem solving.

Before the child can run their program on the Lego Mindstorms robot, the program needs to be compiled into byte code and then downloaded onto the robot.

To initiate the compilation, the child taps the Upload button, which uploads the file to the web service. The successful compilation feedback in the form of a sound and message is provided. Then the student can initiate the downloading of the program onto the robot by tapping Run. To initiate the program the child selects their program on the Lego Mindstorms robot as shown in Fig. 7.

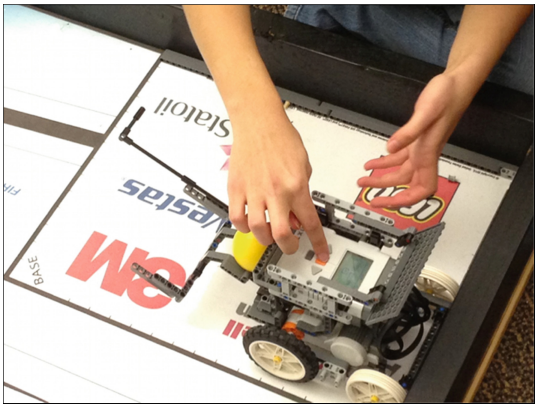

Fig. 7. Child selecting a program on the Lego Mindstorms NXT robot

5 Evaluation

The workflow and the block design were developed incrementally, with user testing at significant checkpoints development. Due to the need to get a diverse sample of children, field tests were conducted as a public project demonstration event that was attended by about 30,000 people. The tasks tested with users included:

1. Creating a simple program to allow the robot to move forward.
2. Edit an existing program to allow the robot to move as desired by the user.

The team tested the user interface with kids of all ages over the course of the day. Children younger than 10, with little to no programming experience were of particular interest to the team. That day 22 children fit this criteria. When a child asked to use the app, a member of the team gave the child a quick overview of the software. Due to the nature of the event, each child did not necessarily complete both tasks. Observations noted the issues with identifying the purpose of the blocks (text, color, icons), how to move blocks around a program, as well as how to add/delete blocks.

Given the nature of the field test, metrics like time to complete task and number of errors were not gathered since the tasks themselves varied. The observations of the children resulted in the following findings:

- Overall the children were able to easily select and move the desired blocks, including adding blocks, moving blocks within the program (including nesting logic), and to the trash.
- The icons selected to symbolize the type of block was clear and legible to the children.
- Some of the background and text/icon color combinations need to be revised for clarity due to legibility issues, especially on a smaller tablet screen (e.g. iPad Mini)
- Some children needed a couple of attempts to move a block within a section of blocks (e.g. within a block of code in an if-then block). An additional source of confusion for some children arose when a block needed to be moved to a part of the program that scrolled down/up the screen.
- As setting variables is new to most children, that aspect of the workflow resulted in initial questions, but setting variables and understanding their intent (e.g. angle to turn) was straightforward.

6 Conclusions and Future Work

The refinement of the system and the addition of new features will move JBrick for iOS forward. Further, user evaluation will provide feedback from a more diverse set of children. Additional features desired include the use of constants to simplify the use of constructs for novice programmers and the means to create custom methods. Additional refinement of the user interface such as the simplification of parameter presentation and server-side are also needed in order to improve usability for the student and the teacher.

An Android version of JBrick is desired, though a rewrite will be needed since JBrick for iOS is a native app. Regardless, high-level design and algorithms can be reused at the conceptual level.

References

1. Johnson, J.: Children, robotics, and education. Artif. Life Robot. **7**(1), 16–21 (2003)
2. Lawhead, P., Duncan, M., Bland, C., Goldweber, M., Schep, M., Barnes, D., Hollingsworth, R.: A road map for teaching introductory programming using LEGO© mindstorms robots. In: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR 2002), pp. 191–201. ACM, New York (2002). doi:10.1145/782941.783002. http://doi.acm.org/10.1145/782941.783002
3. Ludi, S., Reichlmayr, T.: The use of robotics to promote computing to pre-college students with visual impairments. ACM Transactions on Computing Education, 11(3). ACM New York, NY (2011). doi:10.1145/2037276.2037284
4. Sipitakiat, A., Nusen, N.: Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children. In: Proceedings of the 11th International Conference on Interaction Design and Children (IDC 2012), pp. 98–105. ACM, New York (2012). doi:10.1145/2307096.2307108. http://doi.acm.org/10.1145/2307096.2307108
5. Sklar, E., Eguchi, A., Johnson, J.: RoboCupJunior: learning with educational robotics. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 238–253. Springer, Heidelberg (2003)
6. Scratch Screenshot (2013). http://upload.wikimedia.org/wikipedia/commons/5/5e/Scratch_Screenshot_Gravity_Script.png
7. Slany, W.: Catroid: a mobile visual programming system for children. In: Proceedings of the 11th International Conference on Interaction Design and Children (IDC 2012), pp. 300–303. ACM, New York (2012). doi:10.1145/2307096.2307151. http://doi.acm.org/10.1145/2307096.2307151
8. Touretzsky, D., Marghitu, D., Ludi, S., Bernstein, D., Ni, L.: Accelerating K-12 computational thinking using scaffolding, staging, and abstraction. In: ACM Technical Symposium on Computer Science Education (SIGCSE), Denver, CO, March 2013