2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

A Visual Robot-Programming Environment for Multidisciplinary Education

Jennifer Cross, Christopher Bartley, Emily Hamner and Illah Nourbakhsh

Abstract— **Arts & Bots is an educational program that aims to broaden diversity and participation in technology by integrating arts and crafts with robotics. Arts & Bots is a flexible program that can be integrated into in-school and out-of-school programs in many subject areas. This paper describes the visual programming environment developed for Arts & Bots and its goals of low barriers to entry, classroom compatibility, supporting student acquisition of computational thinking skills, and enabling complex robot behaviors. The authors also compare and contrast the programming environment with other popular visual programming environments, namely Scratch, Alice and LEGO NXT-G.**

I. Introduction

Over the past decade, robotics has become an increasingly popular vehicle for supporting student engagement by allowing students to learn about technology both inside the classroom and in extracurricular activities. From the widely available LEGO MINDSTORMS System to popular national robotics competitions, such as US FIRST and Botball, the use of robotics to attract students and instruct them in STEM-related topics has become very common.

In order to provide the benefit of robotics to a broader group of students, who may be uninterested or intimidated by the aforementioned activities, robotics programs and kits such as Robot Diaries[1], Artbotics[2], and PicoCricket [3] were designed to incorporate arts and craft materials into the supplies provided for creating robots. Common craft materials serve a number of purposes in these programs. First, the inclusion of crafts materials is a more gender neutral material than, for example, LEGO components or erector sets which boys often have many years of experience working with prior to their robotics experiences [3]. Crafts are familiar and approachable to most students who have utilized similar materials in previous school projects regardless of their level of interest in technology [1]. The processes that are most commonly associated with arts and craft materials are also intrinsically creative which further supports their use in encouraging creativity in the design process.

Initial pilots with out-of-school groups showed that Robot Diaries was appealing and engaging for students as well as a valuable learning experience[1]. However, self-selection still kept many students from participating in Robot Diaries. In order to engage a wider audience of students, we began to work with teachers to incorporate Robot Diaries into school curricula. The flexibility of the craft materials and robot kit allows teachers to integrate Robot Diaries into a variety of course topics, from poetry to anatomy to history[4]. Now called Arts & Bots, the program has grown into a popular tool for introducing K-12 students to robotics while also increasing engagement with topics from various school subjects. Since 2006, approximately 1000 students and 85 educators have participated in Arts & Bots programs in Pennsylvania, Ohio, West Virginia, and even international locations in Brazil and the United Kingdom.

II. Arts & Bots

The primary goal of Arts & Bots is to increase technological fluency while smoothly integrating into classrooms of diverse disciplines. Technological fluency is the ability to manipulate technology creatively and for one's own use. By focusing on fluency-building activities, which encourage creativity and personal adaptation of technology, Arts & Bots aims to engage diverse student populations with technologies that might otherwise be interpreted as irrelevant to their personal interests.

Arts & Bots uses craft materials, a flexible hardware kit, an interactive software environment, and adaptable curriculum to empower students to create provocative, tangible sculptures with robotic actuation and sensing. The Arts & Bots hardware kits available for sale consist solely of the robotics components that are required for the creation of these craft-based robots. The craft materials required to complement the hardware are selected and provided by the teachers to permit them to adapt the materials to their specific project. Arts & Bots hardware kits include outputs chosen to encourage the creation of compelling narrative robots. These outputs include: DC motors, hobby servos, RGB LEDs, single color LEDs and vibration motors. Sensors, to support interactions with the robot, include: temperature, light, sound level and IR distance sensors and a potentiometer.

These materials permit the construction of a varied set of robots with a high level of morphological diversity. However, the nature of the Arts & Bots microcontroller, called the Hummingbird, allows us to make a number of assumptions about the robots. First, because the Hummingbird is always tethered to a computer, the robots created with the kit have limited mobility and thus are often designed to sit on a flat surface or hang from a wall. Second, while the availability

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (0946825) and the NSF Broadening Participation in Computing program under Grant No. (0940412). This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B090023).

Jennifer Cross, Christopher Bartley, Emily Hamner and Illah Nourbakhsh are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. *Email:* {jcross1, cbartley, etf, illah} AT andrew.cmu.edu.

of craft materials permits essentially limitless mechanical and aesthetic variation, the robots have reasonably uniform electrical systems since each of the fourteen outputs and four sensor inputs is either utilized or not in each robot. This paper describes the programming environment that was developed based on the goals of Arts & Bots and these assumptions.

III. Programming Environment Goals

The Arts & Bots program needed to provide an appropriate software environment to facilitate the programming of robot behaviors and interactions, as well as specifically meet the following goals aligned with the overall purpose and needs of the Art & Bots program.

Computational Thinking - In order to support the primary purpose of Arts & Bots, which is the development of student technological fluency, the Arts & Bots programming environment should strive to support the acquisition of computational thinking skills. To do this, the programming environment should permit the exploration of common computer science practices such as creating reusable code and understanding basic logical structures such as *if-else* conditionals.

Classroom Compatibility - Another key aim of Arts & Bots, and thus the programming environment, is to support the goals of the educator implementing the system in a multidisciplinary fashion. In a language arts class studying poetry, Arts & Bots must encourage deeper student engagement with poetry. In an anatomy class studying muscles, Arts & Bots must help students refine their understanding of the mechanics of the muscloskeletal system. This means that, while the ideal programming environment should have the flexibility to be adapted for different types of projects, it should not be specialized for any singular course topic. Additionally, the programming environment should be compatible with varying computer configurations in order to allow schools to utilize Arts & Bots without the burden of purchasing additional equipment.

Low Barrier to Entry - Following on the intention of fitting seamlessly into many different educational environments, it is also important to minimize the amount of time required to learn how to create programs for Arts & Bots. Doing so minimizes student distraction from the course topic and allows teachers unfamiliar with computer programming languages to instruct students with the confidence required to engage and assist the students. Lower barriers also help to promote technological fluency by supporting students' gradual acquisition of technological knowledge without damaging their confidence by straining their developing competency.

Compelling Behaviors - While it is important to simplify the complexity of learning to program, it is also critical that the programming environment not over-simplify the capabilities of the robots to such a degree that students can no longer achieve their desired narratives. A certain degree of flexibility, and thus complexity, is needed to maintain the desired degree of engagement.

These goals present a challenging design space, where the values of certain goals are frequently in direct opposition to other goals. Being classroom-compatible requires a flexible nature that is aligned with educator goals and does not distract students from that purpose, while encouraging computational thinking could easily cause that type of distraction. Lowering the barriers to entry could be accomplished through simplifying the software's capabilities, but that simplification could potentially reduce the student's ability to create compelling narratives and behaviors.

IV. Implementation

In order to meet these goals, the developers of the Arts & Bots programming environment decided to implement a visual, or graphical, programming language and environment. In order to permit the future expansion of this visual programming environment beyond use with the Arts & Bots hardware, it is called the CREATE Lab Visual Programmer. The CREATE Lab Visual Programmer is an open-source application written in Java and distributed via Java Web Start [5].

Visual programming languages (VPLs) are defined as those that use greater than one dimension for communicating language semantics. VPLs commonly benefit from four features described and defined by Burnett as *concreteness*, *directness*, *explicitness* and *immediate visual feedback* [6]. Concreteness reflects that some aspects of the program are able to be represented as specific objects or values. Directness indicates that the programmer has a feeling of direct control or manipulation and that the mapping from the problem space to the program space is short and clear. Explicitness indicates that some of the semantics of VPLs are explicitly stated in the environment. Finally, immediate visual feedback means that changes to the program are automatically made clear to the programmer.

Based on the goal to lower the barrier to learning to use the CREATE Lab Visual Programmer, the developers implemented a two-step programming environment to provide a scaffold for the creation of robot programs. Building from the idea of a narrative-based robot, these two steps were implemented to reflect the process of designing story scenes and combining these scenes with storyboarding.

The aspect of the CREATE Lab Visual Programmer used to perform the first step is referred to as the Expression Builder, which students use to create static poses or output states for the robot which are called *expressions*, referring to an emotional expression (Figure 1). Static poses are not static in the sense that no robot component is moving, but rather expressions consist of a partial definition of the states of each output on the Hummingbird. For example, one expression could be used to describe when the robot is angry. This expression might set both RGB LEDs to glow red while one vibration motor is turned on to full power to create a growling sound. Another expression on the same robot might be used to demonstrate a happy emotion where the two RGB LEDS could be set to a calm green-blue color, and the vibration motor could be turned off.

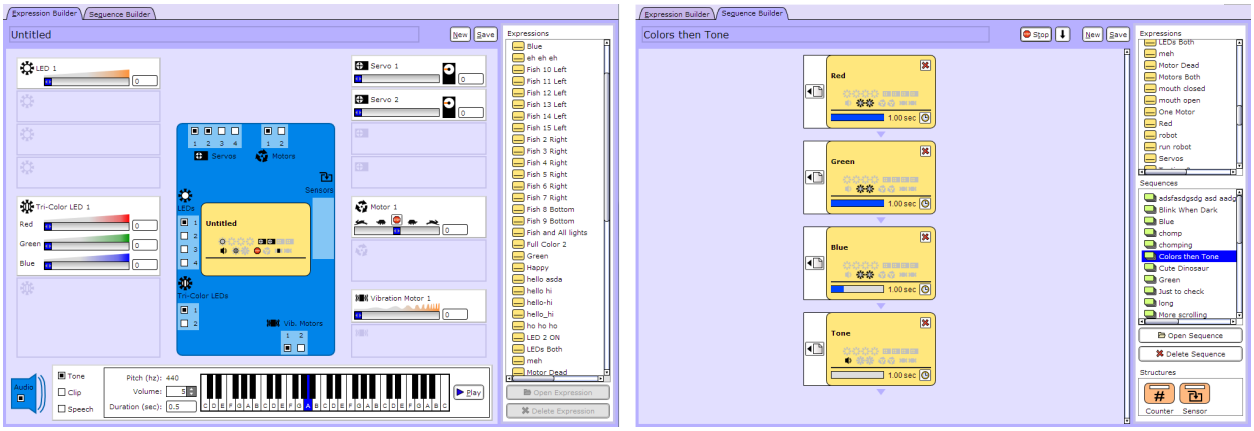

Fig. 1. [Left] A screenshot of the Expression Builder. In the center is an expression block shown in yellow surrounded by a cartoon representation of the Hummingbird in blue. Surrounding the Hummingbird are a number of control panels, which control the attributes of each enabled output. On the right-hand side is a palette where saved expressions are displayed. [Right] A screenshot of the Sequence Builder. Saved expressions and sequences can be dragged from the palette on the right and dropped into the sequence being built in the center.

The aspect of the CREATE Lab Visual Programmer used to perform the storyboarding-step is referred to as the Sequence Builder and is used to define robot behaviors by joining expressions and other program elements into combinations of robot actions that occur over time (Figure 1). These storyboards are referred to as *sequences* since they are most commonly comprised of a time-sequential group of expressions. A basic example of a sequence would be to use the two earlier mentioned expressions and create a sequence causing the robot to alternate from "happy" to "angry" and back to "happy". In order to fully describe this behavior, the time delay between each expression transition must be defined in the program, for example the robot could display "happy" for two seconds, then "angry" for half a second and finally return to a "happy" expression. Once a sequence has been created, it is possible to reuse that sequence as a component, called a subsequence, within a larger or more elaborate sequence. The Sequence Builder also provides access to a list of additional structural elements available for creating more complex sequence behaviors. These include an *if-else* conditional based on sensor readings called a sensor structure and a repeating *for-loop* called a counter structure.

Both the Expression Builder and the Sequence Builder are provided inside the same environment window. The two builders are located in different tabs that are navigable from the top of the window. Both windows also have a persistent sidebar palette that contains a file list where all existing expressions are shown. The Sequence Builder also has a similar list showing existing sequences and the list of available structural elements. The Expression Builder and the Sequence Builder are demonstrated in the accompanying video. Below are presented three key features that the CREATE Lab Visual Programmer implements to meet the stated design goals.

A. Real World Grounding

The software environment emphasizes the use of clear physical metaphors to aid in the process of recognition for students who are programming an Arts & Bots robot for the first time. Following the creation of a robot, each student will be fairly familiar with the functionality and appearance of the physical hardware components of the Hummingbird kits. This existing knowledge is utilized as a foundation for the introduction of the programming task. This idea is closely related to concreteness, in that select components of the interface are designed to imply a direct relationship to physical components and actions.

In order to maximize the concreteness of the environment, the developers also chose to omit the inclusion of variables in the language since the abstract nature of variables is not required to create compelling narratives and would require a level of additional complexity that would distract from most class projects and raise the barrier to entry. The barriers would be especially high for younger students in Piaget's concrete operational stage of development (ages 7-11), who are capable of thinking logically about concrete objects and concepts but are still developing capabilities for abstract thinking[7]. While the inclusion of variables could expand the abilities of the programming environment for older students, there are no immediate plans to add them, as the students in pilot implementations have not demonstrated a need for them.

Upon opening the software, the Expression Builder Hummingbird image dominates the center of the window and is accompanied by prompting text stating "Enable an Output Port". The output ports on the Hummingbird are represented in the software as checkboxes located and labeled in a manner identical to their counterparts on the hardware Hummingbird. Building from this, each output type is also associated with a representative icon that is used on both the hardware Hummingbird and throughout the software (Figure 2). These icons serve to help students recognize either the appearance or function of each output throughout the software. There is also a checkbox that can be used to enable audio from the computer's speakers to be incorporated into an expression.

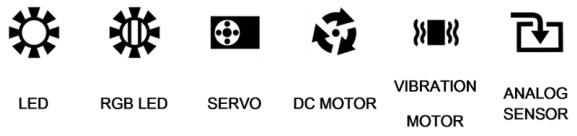

Fig. 2. The icons used to represent Hummingbird outputs and inputs.

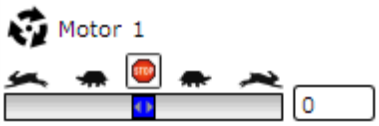

Fig. 3. An example motor control panel has a slider bar that has a stop button located above its center location. This button is used to set the motor speed to zero. On either side of the button are silhouettes of tortoises and hares moving away from the center that indicate that the slider is for selecting speed and direction.

The action of checking an output port checkbox is a metaphor for plugging a component into the Hummingbird. Once checked, a control panel, i.e. a region representing and controlling the attributes of that output, is shown in the Expression Builder. The control panels contain at least one slider bar that is used to set the state of that specific output (Figure 3). Additional visual cues are also used to indicate what result to expect from moving the slider. Once the slider position is altered, the new output setting is immediately reflected onto the physical robot. For the computer audio control panel, instead of a slider, a keyboard is provided to allow the selection of tones or the student can view a menu of sound clips from which to select a sound effect.

Similarly, the metaphor of making a storyboard for a narrative is the basis of sequence creation. By representing expressions and subsequences as colorful blocks with titles and icons representing the function, the sequence is able to incorporate these elements in a way that is visually similar to the frames of a storyboard. These elements are stacked vertically in a snap-in-place sequential arrangement. However, the use of sensor structures, which can be set to act as *if-else* conditionals or *while* loops, can be used to alter the control flow (Figure 4).

B. Live Feedback and Debugging

Another aspect that eases the process of learning to program using the CREATE Lab Visual Programmer is that there are no syntax errors that can be made in the environment. Each control panel in the Expression Builder is designed to have a one-to-one relationship with an output on the Hummingbird. In using the control panels to set the output value, it is not possible to select an out of range value. If a number is typed in that is out of range, it is adjusted to be either the maximum or the minimum allowable value. This means that every expression is valid and executable. The expressions are then combined using the Sequence Builder with a click-and-drag interaction that allows each element to be placed in the active sequence by snapping them into a valid position. Syntax errors are also not possible when implementing sensor loops in a sequence. Each sensor loop

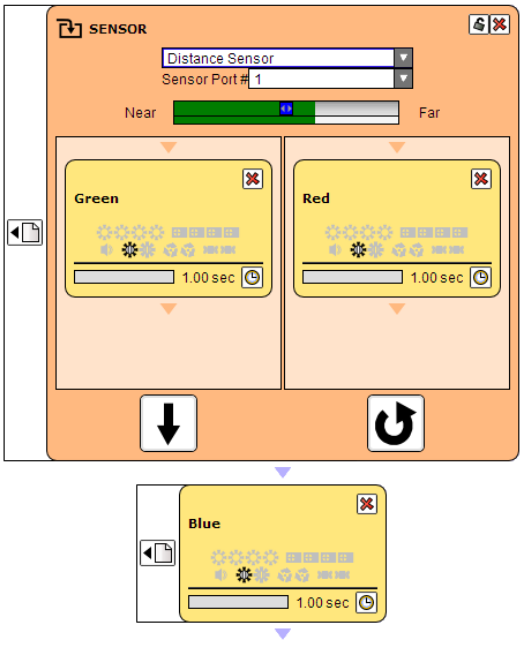

Fig. 4. A screenshot of a sensor loop. Two drop-down menus are provided to allow for the selection of a sensor type and a port number. The green value bar beneath the menus shows the current live sensor reading as well as the position of the threshold, in blue. If the sensor value is greater than the threshold, the right hand pane is executed. If the sensor value is less, the left hand pane is executed. The black arrow at the bottom of each pane allows the programmer to decide whether, following the execution of the pane, the program continues to the next element in the sequence or if it loops back to check the sensor value again. This allows for the implementation of *if-else* statements, *while* loops and infinite loops containing *if-else* statements.

has two drop-down menus, which allow for the selection of a sensor type and a valid sensor port number. The sensor threshold is also limited by its slider such that the threshold is always within the range of the sensor values.

In order to assist in the debugging of semantic errors, or errors where the program created does not do what the student expects, the CREATE Lab Visual Programmer features live visual feedback. While creating expressions, the Hummingbird's physical outputs immediately update to match every change made on a control panel. This allows for a rapid feedback loop where the student adjusts an output value, gauges the effect that the adjustment had on the physical robot and continues to modify the expression values until the physical robot's pose meets her needs. Similarly, the continuous, live readings from the sensors in the Sequence Builder make setting and testing threshold values very straightforward. Once a threshold is selected, the student can vary the sensed environmental condition, i.e. temperature or light level, to ensure that the sensor reading is above and below the threshold during the times that she expects.

Debugging semantic errors in sequences is also aided by the explicitness of the sequence execution flow. Each program element is directly above and connected by an arrow to the element that follows it during execution. It is straightforward to interpret the sequence of actions that will

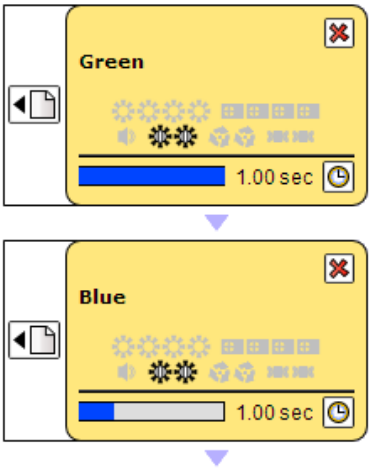

Fig. 5. Sequence execution flow can be viewed in real-time as the sequence executes.

Fig. 6. The "Human Seasons" poem-bot is seen from four sides. Shown clockwise from the top-left: Spring-Childhood, Summer-Adulthood, Fall-Maturity and Winter-Death. In the image of Winter-Death, the Hummingbird and other hardware components are visible inside of the robot's base.

occur when the program is executed simply by following the arrows. The time delay following an expression that prevents the immediate execution of the following program element is also explicitly written in the bottom portion of each expression block. Along with the specified delay value is a progress bar that, while the sequence is executing, fills as the delay passes (Figure 5). Once an expression's delay is completed, its associated delay progress bar remains filled until the sequence completes execution or that particular expression is reset and re-utilized in a sensor or counter loop structure. The sensor and counter loop structures also both have similar highlight indicators that show which pane of the sensor structure is executing and show a progress bar of how many repetitions have been completed by the counter loop. This filling and highlighting action helps students recognize which expression or program element is executing while the program runs, helping with both timing issues and determining where in the program the error occurs. Once the error location is identified, it is possible to edit sequences even while they are running, allowing for immediate corrections which can lead to experimentation to correct the problem. In order to further assist first-time programmers, who may not yet know how to set the time delays, each delay defaults to one second so that, on the first run, each expression lasts long enough to be seen on the physical robot and tracked by following the highlights.

C. Incremental Complexity

In order to meet the opposing goals of having a low barrier to learning and having the level of complexity required to create a compelling robot that incorporates computational thinking, the CREATE Lab Visual Programmer was designed to be highly scaffolded and supportive to novices and also to allow more experienced users to push the limits of the environment to create more elaborate robot behaviors.

The scaffolding is primarily reflected in the use of the two-step programming process for making expressions separate from sequences. This separates the process of finely controlling individual outputs from the high-level design of the overarching robot action. In doing so, the CREATE Lab Visual Programmer enforces the process of creating low-level functions to be used in the primary program. This segmentation also helps to lower the floor even further for the youngest users, such as those in kindergarten, who are not able to plan the creation of a custom sequence from start to finish. One approach is to let the young students tinker with the Expression Builder to customize the robot's pose. Another approach is to have the teacher create a number of simple and clearly labeled expressions that young children can assemble into their own sequences.

Beyond the benefits of the two-step programming approach, many other features of the CREATE Lab Visual Programmer permit the gradual integration of more complicated computer science concepts. At the most basic level, a student can create expressions that individually define the complete output state of the robot and then combine these into a sequence. These basic sequences are fully capable of providing a robot with an interesting narrative behavior, however many improvements can be made by utilizing other features of the environment. When the student becomes comfortable with expressions, the counter structure can be introduced as a way to make repetitive sets of expressions easier to modify and interpret. Next, the use of sensor structures can provide the added dimension of interactivity with the robot's surroundings. As sequences grow increasingly complex, the idea of creating small pieces of testable, reusable code in the form of subsequences can also become a valuable skill. In the following section, some of these optional features are implemented in a piece of example code written by a group of eighth grade students.

V. Case Study

The following case study is an unmodified example of a robot built and programmed by a team of four 12- and 13-year-old students in an eighth grade Language Arts class studying poetry. As part of their class, the students were

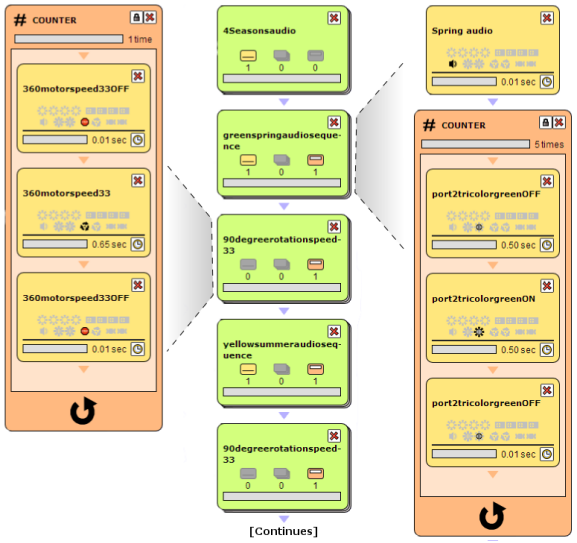

Fig. 7. The main sequence created for the "Human Seasons" poem-bot is shown center. To either side are views of the contents of the subsequences; to the left is the subsequence for rotating the platform 90 degrees and to the right is the subsequence for the presentation of the Spring-Childhood stanza.

asked to build robots to present the message and meaning of a specific poem. The process of designing and decorating the poem-bots allowed the students to explore the imagery of the poems, while the process of creating sequences help them explore the importance of time in dramatic presentations. The robot designed, constructed and programmed by the team is pictured in Figure 6. This robot was created to express the poem "The Human Seasons" by John Keats which associates the stages of a human life with the four seasons of the year.

The robot consists of two main structural components: a base that contains all of the Arts & Bots hardware; and, mounted on top of a DC motor, a round turntable divided into four sections. Each of the sections is decorated to be representative of both a season and the associated stage of life. As the platform rotates to display each season, the robot also activates an RGB LED spotlight to highlight the season facing towards the audience. Additionally, the robot's sequence contained expressions with audio clips of the students reading each stanza of the poem that played when the sequence was executed. As each stanza is read, the robot moves to the correct season and illuminates that season with an appropriately colored light.

The sequences and expressions that these students created for their robot demonstrate a good understanding of many of the CREATE Lab Visual Programmer features mentioned above. By looking at the program that they created, it is also possible to conclude that the students gained an appreciation for a number of the computer science concepts that Arts & Bots aims to encourage.

A. Computational Thinking

In the example shown in Figure 7, the students designed their primary sequence to be composed entirely of subsequences, shown in green. The third subsequence, titled "90degreerotationspeed-33", is designed to rotate the platform 90 degrees in order to turn to the next season. This subsequence only uses expressions, which provide commands to the DC motor, unlike another style of expression that would control the entire robot. This demonstrates that the subsequence was not designed to be used exclusively during a single transition; instead, it shows that the students have put effort into creating reusable code. In the main sequence, we can see that they do indeed reuse this subsequence before each new season. This saved them the time required to recreate this behavior for each season and permits them to quickly modify all of the rotation subsequences if they would later decide to change the speed of rotation. This is all the more valuable when using a DC motor since the speed and timing aspects are critical to get right when the robot has no feedback and the rotation is based on dead reckoning.

The other subsequence shown is used to perform the Spring-Childhood stanza. The first expression starts playing the audio performance. Since the delay for this expression is set to .01 seconds, the counter structure begins to execute at approximately the same time as the start of the audio. The counter structure is then used to blink the RGB LED while the audio is played. The use of the counter structure here is a good indication that the students understood the value of using a programmatic loop when performing a repetitive task. One benefit to creating the season performance in its own separate subsequence is that the students were then able to run and debug each part of their program individually. This means that it was possible to make improvements to each season without needing to perform the full poem each time.

The application of computer science principles demonstrated by these sequences and expressions is a good indicator that the CREATE Lab Visual Programmer succeeded in accomplishing the goal of increased computational thinking. Perhaps more importantly than the gains deduced, students in the class also self-reported changes in their relationship with technology because of the project. One student stated simply that "it made me confident about tecnology[sic]" while another said "it made me feel more like I knew things about it."

B. Low Barrier to Entry

The experience of using the CREATE Lab Visual Programmer also caused students to change their perceptions about the difficulty of programming a robot. One student was able to build the skills required to program the robot and reflected on the process by saying that "Programming is very challenging but once you get used to it, it's easy." Another student found the process to be easier than he expect and explained that "I learned that even though programming looks difficult it is actually easier than it seems." This ease of use and the process of building abilities to meet the challenge of programming is indicative that the CREATE Lab Visual Programmer successfully lowers the barrier to robot programming.

TABLE I
PROGRAMMING LANGUAGE COMPARISON

	Alice	**Scratch**	**LEGO NXT-G**	**Visual Programmer**
Character Customization	Structured	Flexible	Structured	Flexible
Behavior Programming	Structured	Flexible	Flexible	Structured

C. Classroom Compatibility and Compelling Behaviors

Students working on the poetry-bot project reported that they thought that the Arts & Bots project aided their appreciation of poetry. One student in the class explained that "Poetry can sometimes be hard to understand but using robotics and giving you a visual can help you understand it. [sic]" The teacher was able to smoothly integrate the project in with her curriculum and get the CREATE Lab Visual Programmer installed and operating on all of the school-provided laptop computers. During the final presentations of the project, the class displayed eight unique poem-bots, each of which was able to present its poem with an expressive visual display. The experiences of this class are similar to other classes that have integrated Arts & Bots into their curriculum. This suggests that the CREATE Lab Visual Programmer successfully achieved the goals of classroom compatibility and enabling compelling behaviors.

VI. COMPARISON TO SIMILAR ENVIRONMENTS

There has been a great deal of work done on the development of programming environments that are suitable for first time programmers. Three of the most popular visual programming environments currently used in schools and extracurricular programs are Alice[8]; Scratch [9], [10]; and LEGO MINDSTORMS NXT-G[11]. These languages, as well as the Visual Programmer, all permit the completion of a similar task: programming behaviors of virtual or robotic characters through a graphical programming environment. The Visual Programmer is distinguished from these languages, however, in considering how scaffolding structure is balanced against flexible customization throughout the programming process. An overview of these differences is shown in Table I.

A. Alice

Alice is a language that allows students to create 3D interactive graphics, including animations and games. To create a program, students use an object-oriented approach to select predefined 3D characters to add to the scene of their animations. Using context menus, it is then possible to select from a number of provided character functions or actions. This selection-based approach provides structuring which allows novices to explore the capabilities of the characters without existing knowledge. This is similar to the scaffolding utilized in the CREATE Lab Visual Programmer's Expression Builder to introduce the capabilities of the various outputs using the control panels as visual indicators of actions. However, 3D Alice characters are difficult to customize, while the Visual Programmer permits much more flexibility in character creation by allowing students to build unique robots.

A variation of Alice, called Storytelling Alice[12], leveraged the idea of storytelling to create a motivation for the creation of programs using the object-oriented approach. Students involved with the storytelling tasks were found to be more inclined to take future Alice courses and would spend extra time to work on their stories over non-Storytelling Alice. Arts & Bots and the CREATE Lab Visual Programmer use a similar approach for motivation, focusing on the creation of expressive robots and narrative programs.

B. Scratch

Scratch is a language that students can use to create interactive multimedia projects, like games and interactive stories, which incorporate 2D graphics, audio and video components. Scratch also emphasizes the importance of flexibility in character creation by supporting the student in designing custom 2D characters called *sprites*. Similarly motivated, the creation of unique physical robots in Arts & Bots is not just encouraged but mandatory since the Arts & Bots hardware components are entirely meaningless until a robot is designed and built. Programs in Scratch are created by snapping together command blocks that can control the actions of sprites. While the method of assembling programs from command blocks in Scratch is similar to how program elements are combined into sequences with the CREATE Lab Visual Programmer, command blocks focus on finer controls of individual aspects of the sprites while expressions are high-level state settings for the entire robot. Using command blocks Scratch also permits more abstract programming concepts like variables and data manipulation that are not possible with the CREATE Lab Visual Programmer. These differences allow Scratch programs to have greater flexibility than CREATE Lab Visual Programmer expressions and sequences, however learning these features can be time consuming, making Scratch more suited to classrooms solely focused on technology education.

Finally, Scratch also incorporates debugging and live feedback features that are similar to those that are implemented in the CREATE Lab Visual Programmer. The building block nature of Scratch command blocks prevents the presence of syntax errors since non-compatible commands cannot be combined. Scratch, like the CREATE Lab Visual Programmer, also highlights the code as it is executed and allows for live editing of programs at runtime to help identify and correct for semantic errors.

C. LEGO MINDSTORMS NXT-G and RoboLab

LEGO MINDSTORMS NXT-G and its predecessor RoboLab[13] are two similar visual programming environments that are used to program LEGO MINDSTORMS NXT robotics kits in schools. These environments are both based on the popular dataflow programming environment, National Instruments LabVIEW, which is programmed by defining

how data and signals flow between various functional blocks. Similar to the CREATE Lab Visual Programmer, both of these environments are specifically designed to control physical robots built by the students. However, NXT-G has more computational capabilities including variables, mathematical functions and logic constructs.

LEGO MINDSTORMS has a more structured robot character creation process than that of Arts & Bots. Component uniformity makes the construction of LEGO robots a more defined process, and possible to accomplish by following illustrated directions. At the same time, LEGO components are comparatively more expensive than craft materials; this deters most users from modifying LEGOs with non-LEGO components. This leads to a trade-off where MINDSTORMS robots are less customized and less personal than Arts & Bots robots. MINDSTORMS robots are also limited to three outputs whereas Arts & Bots robots can use up to fourteen. Since students learning to use Arts & Bots will likely be more experienced and confident using craft materials than writing programs, a structured programming experience and flexible, personalized construction is more suited to the intended goals of Arts & Bots.

VII. Future Directions

In the future, further refinements are planned to increase the functionality of the CREATE Lab Visual Programmer and allow for easier integration with different classroom environments. One of the most exciting features planned is to allow programs to be exported from the CREATE Lab Visual Programmer in the form of ready-to-execute Java code. By allowing code to be exported, the CREATE Lab Visual Programmer will ease the transition for students who are looking to graduate to a more powerful, flexible, and widely used programming language.

Meanwhile, the Arts & Bots project will continue to focus on the implications of robotics in multidisciplinary classrooms. The project is currently growing the number of involved educators and improving the quality of connections that are being made between Arts & Bots and existing curricula through these teacher collaborations. Through analysis of future classroom observations and future student assessments, as well as the evaluation of survey and interview data collected in current pilots, Arts & Bots and the CREATE Lab Visual Programmer will be used as catalysts for exploring what factors are important to consider when implementing robotics projects within different school disciplines.

VIII. Conclusions

The CREATE Lab Visual Programmer was created in order to fulfill the goals of the Arts & Bots program by supporting the creation of complex robot behaviors and encouraging students to develop technology fluency. In order to do this, the environment needs to have low barriers to entry, be compatible with classroom implementation and support student acquisition of computational thinking skills. The design of the CREATE Lab Visual Programmer has achieved these goals by focusing on the key factors of incorporating real world grounding, live feedback, semantic debugging aides and the ability to increment the complexity of software as the student gains skills and confidence. Classroom observations and student responses have demonstrated positive gains that will be explored further in the next stage of the Arts & Bots project.

Acknowledgments

We gratefully acknowledge the many people who have contributed to the development of Arts & Bots including: many members of the CREATE Lab, the educators with whom we have worked, the June Harless Center for Rural Educational Research and Development, and Tom Lauwers of BirdBrain Technologies.

References

[1] E. Hamner, T. Lauwers, D. Bernstein, I. Nourbakhsh, and C. Disalvo, "Robot Diaries : Broadening Participation in the Computer Science Pipeline through Social Technical Exploration," in *Proceedings of the AAAI Spring Symposium on Using AI to Motivate Greater Participation in Computer Science*, Stanford, CA, USA, 2008.

[2] H. A. Yanco, H. J. Kim, F. G. Martin, and L. Silka, "Artbotics: Combining Art and Robotics to Broaden Participation in Computing," in *Proceedings of the AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, Stanford, CA, USA, 2007.

[3] N. Rusk, M. Resnick, R. Berg, and M. Pezalla-Granlund, "New pathways into robotics: Strategies for broadening participation," *Journal of Science Education and Technology*, vol. 17, no. 1, pp. 59–69, Oct. 2007.

[4] E. Hamner and J. Cross, "Arts & Bots: Techniques for distributing a STEAM robotics program through K-12 classrooms," in *Proceedings of the Third IEEE Integrated STEM Education Conference*, Princeton, NJ, USA, in press 2013.

[5] CREATE Lab. Code Repository: create-lab-visual-programmer. Accessed: February 8, 2013. [Online]. Available: http://code.google.com/p/create-lab-visual-programmer/

[6] M. Burnett, "Software engineering for visual programming languages," in *Handbook of Software Engineering and Knowledge Engineering Vol. 2*, S.-K. Chang, Ed. Singapore: World Scientific Publishing Company, 2001, vol. 2, pp. 77–92.

[7] D. Coon and J. O. Mitterer, *Introduction to Psychology: Gateways to Mind and Behavior*, 12th ed. Belmont, CA, USA: Wadsworth Cengage Learning, 2008.

[8] M. Conway, S. Audia, T. Burnette, D. Cosgrove, and K. Christiansen, "Alice," in *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*. New York, New York, USA: ACM Press, 2000, pp. 486–493.

[9] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch Programming Language and Environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–15, Nov. 2010.

[10] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver, "Scratch," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.

[11] The LEGO Group. LEGO MINDSTORMS Education: NXT User Guide. Accessed: September 17, 2012. [Online]. Available: http://www.legoeducation.us/

[12] C. Kelleher and R. Pausch, "Using storytelling to motivate programming," *Communications of the ACM*, vol. 50, no. 7, p. 58, Jul. 2007.

[13] B. Erwin, M. Cyr, and C. Rogers, "Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school," *International Journal of Engineering Education*, vol. 16, no. 2, pp. 1–11, 2000.