Alice2: Programming without Syntax Errors

Caitlin Kelleher, Dennis Cosgrove, David Culyba, Clifton Forlines, Jason Pratt, and Randy Pausch
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15207
caitlin+@cs.cmu.edu

ABSTRACT
Alice2 is a programming environment designed for teaching programming through building 3D virtual worlds. Based on feedback from user tests, we have created a drag and drop programming system that allows users to experiment with the logic and programming structures taught in introductory programming classes without making syntax errors. While similar programming environments for beginners allow users to experiment with only a few programming concepts, Alice2 allows users to experiment with conditionals, count loops, while loops, variables, parameters, and procedures.

KEYWORDS: Novice Programming Environments

INTRODUCTION
Learning to program is hard. Beginners must learn to find structured solutions to problems, express those solutions in a rigid, formal syntax they must memorize and mechanically enter, and learn to understand the behavior of the running program. The combination of these three tasks can be overwhelming and frustrating for many beginning programmers. The previous version of Alice helps users to understand their running programs in two ways: Alice displays program state visible at all times in a 3D virtual world and animates all state changes [3]. While users generally understood the behavior their programs, user testing of Alice revealed that the necessity to enter programs by typing was frustrating for beginning programmers: 65% of users cited the need to type and 45% cited difficulty with remembering the syntactic details as one of the worst three things about Alice [3]. For these users, typing was a dominant problem in learning to program.

Alice2 builds on Alice and addresses the problems with typing programs by adding a no-typing interface for building and editing programs. Users drag and drop tiles with the names of commands and objects on them to build programs that control the behavior of 3D virtual worlds, often short animated movies and simple interactive games. Unlike many no-typing programming interfaces, Alice2 is not a toy; it includes the programming constructs found in general-purpose languages such as Java and C++ and a simple form of parallel programming. Classes at Carnegie Mellon have demonstrated that it is possible to build relatively large (3000 line) programs in Alice2.

PROGRAMMING IN ALICE
There are five regions in the Alice2 interface: 1) the scene window, 2) the object tree, 3) the object details area, 4) the animation editing area, and 5) the behaviors area. Throughout the interface, elements that might be used in a program (commands, programming constructs, 3D objects, objects' properties, and variables) are tiles that users can drag and drop into animations they are creating. When users drop a command tile requiring parameters, Alice2 displays menus with valid parameter choices for users to select from. By dragging and dropping programmatic tiles, users can learn to express solutions to problems as programs without needing to memorize syntactic rules and command names.

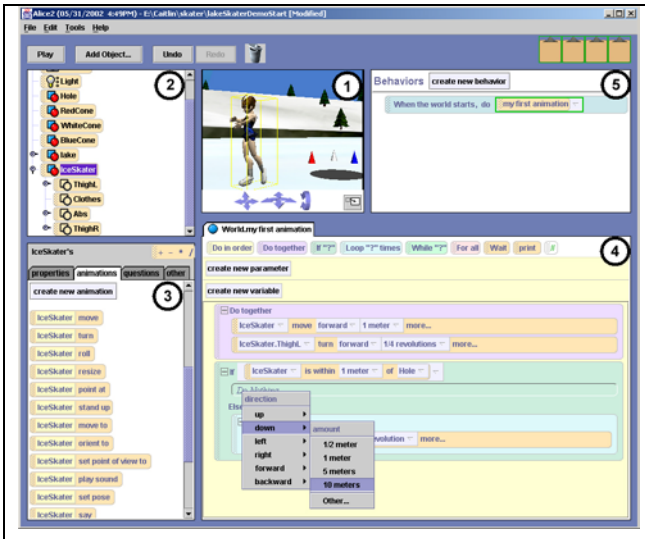

Figure 1: The Alice2 interface: the scene window (1), object tree (2), object details area (3), animation area (4), behaviors area(5)

1) Scene Window
Users can add objects to the scene by dragging them from a web-based object gallery. The scene window also allows users to position and resize objects to create a beginning state for their worlds. In informal observations, we noticed that students often peruse the gallery for ideas on what to build. In response, the Alice2 gallery includes a wide variety of 3D objects and characters including several themed groups such as Japan, Ancient Egypt, a futuristic science fiction environment, and others. We do not currently provide a way for users to create their own objects. Although making objects can be fun for users, it introduces additional interface complexity and does not help users learn to program. However, we are working with the Right Hemisphere

Corporation to allow students to load 3D models from the web in a variety of formats.

2) Object Tree

The object tree provides a hierarchical list of tiles representing objects and subparts of objects in the world. Users can select the object they would like to work with by clicking on the name of that object in the object tree.

3) Object Details Area

The object details area provides additional information about the selected object: the object's properties (such as color, opacity, etc), the animations the object can perform, and questions the object can ask about the state of the world. Users can drag animations and questions from the object details area and drop them into animations.

4) Animation Area

The animation area allows users to view and edit their animations. Each open animation is given a tab in a tabbed pane so that users can have several animations open at a time. Users can drag animations and questions from the object details area and program control structures such as If/Else, While, Loop, and Do Together from the top line of each animation editor. In addition, users can create parameters and local variables for their animations.

5) Behaviors Area

The behaviors area allows users to attach responses to a variety of events in the world, such as the world starting to play, a mouse click on a particular object, and two objects in the world moving to within a threshold distance of each other. The behaviors support an event-based style of programming and can be used to create interactive worlds.

Saving Worlds and Characters

In the previous version of Alice, users could only save full worlds. Consequently, many users ended up writing the same basic methods (e.g. walk) for a given character several times. In Alice2, users can save both worlds and individual characters with animations and behaviors, facilitating both the reuse of animations for particular characters and sharing characters with friends.

RELATED WORK

The Cornell Program Synthesizer was a text-based editor that greatly reduced the number of syntax errors users make by using command templates for commands and programming structures[5]. Users typed a command code like the IF-statement below to add a template at the current cursor position. Users filled in the conditions and statements either by typing or inserting additional templates.

```
IF (condition)
        THEN statement
        ELSE statement
```

Figure 2: Cornell Program Synthesizer IF template

While the Cornell Program Synthesizer allows users to explore the common programming structures without making as many syntax errors, users can still make syntax errors while typing conditions and assignment statements. In addition, users are required to memorize the command codes for inserting templates[5]. The Alice2 animation editor allows users to experiment with the same programming constructs, but uses tiles to reveal the available functionality of the system. Alice2 also uses menus and tiles to prevent the user from making syntax errors.

LogoBlocks[2] and Squeak e-Toys[1], other programming environments for beginners, use graphical objects labeled with words, but neither provides the variety of programming constructs available in Alice2: LogoBlocks includes conditionals and count loops[2]; e-Toys includes conditionals, variables, and procedures[1]; Alice2 includes conditionals, count loops, while loops, variables, parameters, and procedures.

CONCLUSIONS AND FUTURE WORK

Early, informal classroom experience indicates that the Alice2 interface allows novices to learn the logic and control structures of programming without encountering the frustrations associated with entering programs by typing on a keyboard. We intend to use Alice2 to demonstrate that learning the logic and control structures of programming followed by the syntax is a more effective strategy for learning programming, particularly in introductory college courses. For this audience, the process of transferring from Alice2 to a general-purpose language such as Java or C++ is critical. To this end, we have created added the capability to render programs in Java-style syntax. We intend to evaluate how using this Java-like syntax before moving to Java will ease the demands of learning Java syntax.

In addition, we would like to use Alice2 to give a positive first programming experience to middle school girls. It is clear that removing the mechanical difficulties of programming is necessary but insufficient to interest more middle school girls in programming. One possibility we intend to explore is making Alice2 more character-centric. Since girls tend to focus on the characters involved in games [4], developing interesting characters that can interact with each other may be a more compelling activity for this audience.

REFERENCES

1. Squeak Etoys: Background and Tutorials. Available at: http://www.squeakland.org/author/etoys.html

2. Begel, A., "LogoBlocks: A Graphical Programming Language for Interacting with the World." Boston, MA, 1996, MIT.

3. Conway, M., "Alice: Easy-to-Learn 3D Scripting for Novices." School of Engineering and Applied Science. Charlottesville, VA, 1997, University of Virginia**:** 242.

4. Inkpen, K., et al., "We Have Never Forgetful Flowers in Our Garden: Girls' Responses To Electronic Games." *Journal of Computers in Math and Science Teaching*, 13(4),383-40,1994.

5. Teitelbaum, T. and M. McIlrow, "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment." *Communications of the ACM*, 24(9), 563-573, 1981.