Alice: Reinventing Introductory Computer Science

Jason M. Grant
Department of Computer Science & Electrical Engineering
University of Maryland-Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
Collaboration Technology Research Lab
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309

ABSTRACT
One of the most complicated problems in the computer science world is teaching introductory computer science classes to novice computer users. Researchers at Carnegie Mellon University have developed a programming language called Alice to help bridge the gap between introductory programming and other object-oriented languages. In the fall of 2006, the University of Colorado in Boulder will begin to implement this language into their introductory computer science classes. This study served as a preparation for the course, exploring the main features to help create homework problem sets for that course. Also throughout the study, I used Alice to explore its possibilities in working with graph theory. In summary, I concluded that it would be possible to teach Alice in the course of a semester. However, there were still a number of problems that I encountered that could cause difficulty while teaching this language.

General Terms
Alice Programming Language

Keywords
Computer Science Education, Introductory Programming

1. INTRODUCTION
Alice, a fairly new programming language created by the Stage3 research team at Carnegie Mellon University, deals with a common issue in the computer science field: retention of only about half the students in introductory computer science classes. Many computer science students are attracted to the field through video games, animated films, and multimedia; however, most computer science students do not program on high levels such as this until late in their coursework[4]. Unlike many other computer programming languages that are used for high level data processing and computation, the Alice programming language was made specifically as a teaching utility to introduce the concepts of programming to novice users [2]. Thus, Alice was not created to replace other programming language, but to be a bridge for beginners into the world of advanced programming. This software is also aimed to ignite interest in people toward the field of computer science, especially underrepresented groups such as women. Instead of traditional computation, some of the ideas Alice uses to encourage programming are three dimensional graphics and storytelling [2]. Although the language is simple to learn, it still allows advanced users to create complex programs. Finally, Alice, which is built on the Java platform, helps transition users into other object-oriented languages such as Java and C++ by teaching good programming habits [4].

Some of the concepts that make the Alice language better for teaching than other languages are its lack of syntax errors, objects are available to import into program and come with a list of properties and commands for that objects, and the material is organized. Also, the program is built-in with interactive tutorials that guide you through the process of programming with Alice. Included also, is a set of example programs so that a user can see not only the end result, but the code that was used to create the game or story as well.

Alice is known as a fifth generation programming language, which means it is a program based on solving problems using constraints to the program instead of using constraints given by an algorithm written by a programmer[7]. Alice is also classified as an object-oriented language. These types of languages are comprised of individual units, known as objects, that can act on each other as opposed to a language that only has a collection of functions, which simply give instructions to one another[4]. Object-oriented programming languages are known to have greater flexibility and maintainability for writing programs. These types of languages are often used in large-scale software engineering. For example, the language used to operate an aircraft can contain around four million lines of code [3].

As a professor of computer science at the University of Colorado, Dr. Clarence (Skip) Ellis, will be teaching Alice as an introductory computer science language. This prompted us to explore Alice and the majority of the features, My objective was to find out the basic concepts that were involved in the program, some novel features that made Alice
different from other programming languages, and some features of Alice that would make the language fun to teach in the classroom. Also, since the language is fairly new, I was responsible for finding errors and bugs that might make it difficult for a student to program in this language.

To properly evaluate the program, we started out by reading the textbook, watching videos and completing tutorials in Alice so that we would have some background information on the program before evaluating. Once I began to develop some basic concepts of the program, I experimented with different options and functions. Then, I started to go through various exercises in the textbook to learn different task and incorporate what I had learned into practical knowledge. After completing exercises, I pointed out selected exercises that I thought could be useful as homework problems for students. Also during this time, I was to make a note of features that I thought were important, and also the ones that I thought were not so important. The ones that were of importance would be used to develop a set of homework problems for the class throughout the semester.

The remainder of the paper is organized as follows: First, I will introduce the features and functions of Alice that we think will be most useful in the class. In the following section, I will discuss the sets of homework problems that we developed over the course of our study. The next section describes how Alice was used to work on graph theory problems. And finally, the paper concludes with and overview of Alice, thoughts, and conclusions.

2. FEATURES AND FUNCTIONS

Alice keeps users from making syntax mistakes by providing drag and drop items throughout the programming window. Therefore, even when a user begins to write a simple program that just requires the computer to move down a list of commands, the user will select from the methods editor window an icon that's "do in order". This tells the computer to do the task in the specified order that the user has listed the items. Another similar command to the "do in order" command is the "do together" command. The "do together" command allows multiple tasks to be done at the same time, which is very hard to find in other programming languages. Most languages only allow users to do commands one at a time, or do two commands over and over again very quickly, which makes them appear to be moving together. Next to these to-commands are the program's control statements, "if/else", "loop", "while", "for all in order", "for all together", "wait", "print", and lastly a commenting icon. For the programs I wrote, the commenting, "wait", "while", and "if/else" statements were the ones most commonly used.

When inserting objects into your new program or world, the name Alice gives its programs, the user is given a set of commands that are predefined for each object; these commands are called methods in Alice. For instance, a ball has methods such as move, turn around, turn to face, and roll. To implement these commands, a user simply drags the icon from the methods menu and drops it into the program editor. Also, every object comes with a set of predefined functions. For example, the ball will include functions dealing with its proximity to another object, functions about its size as compared to others objects, its spatial relation, and its point of view. Functions such as proximity are very helpful when writing programs when a need an object to move to another object but aren't sure how far away the two objects are.

There are many other features in Alice that are not completely necessary to write a program, but they can make programs and stories more interesting to watch and create. First is the fog function. When a user displays the properties of the entire world, he has the opportunity to change the amount of fog and fog density on screen. This can be extremely helpful when a user wishes to create nighttime or early morning scenes. Another feature that I found to be quite interesting was the addition of music and sound to the program. For example, if a user develops a program that has someone playing an instrument, a user can import a sound clip that he has on his computer and play it while the animation is playing.

Alice has many functions that can be helpful when creating a game. When creating a game, it can be particularly helpful if a player does not know what event will happen after another. Therefore, Alice's built in random number generator allows the creator to select a maximum and minimum number, after which Alice will generate numbers within that range. Also, Alice allows users to place hidden objects in the world, and allows them to be displayed when the programmer wishes for them to be displayed. For example, a person wishing to develop a game in Alice might want to have a win or lose sign hidden until the end of the game, and then either one or the other would show. Also, the hidden objects can be helpful when moving more than one object in relationship to the same object, especially if you do not want the object to show on the screen.

Because Alice could not create every movement possible for every object, Alice allows users to create their own methods for different objects. Take a skater for example, Alice allows the skater to move in all directions; however, to replicate the actual movement of a skater, you have to take into consideration that the upper body might move forward as the skaters legs move forward and gently lean backwards as the skaters legs move together. Alice allows users to create their own commands to produce skating movements by moving body parts together to create a more fluid motion. Once these new methods are created, Alice lets the user save the object, such as the skater, to a file so that the new skater with its current movements can be used in other programs.

Another unique feature of Alice is the creation of parameters and variables. Alice allows the user to create a parameter or variable and then saves it as an icon. Therefore, when the variable or parameter is used in the program, the user does not have to worry about misspelling or typing in the wrong name. Also, if a user decides to change the name of a variable or parameter, the name is changed everywhere throughout the program.

The vehicle command was other feature in Alice that I commonly came across. The vehicle property allows an object to move in reference to another object. So if a sheriff was holding a gun in his hand, a user would normally make the sheriff's hand a vehicle by which the gun is moving. This allows users to type only one command to move the hand.
and the gun rather than typing a command to move the gun and another command to move the hand.

In upcoming months, Dr. Ellis will be leaving to go to Africa to teach a computer science class there. One of my tasks was to develop something in Alice to help in teaching computer science to African students. Dr. Ellis and I decided that we would use he-builder and she-builder to create African characters; however, these features are very limiting in creating people. There is a very limited amount of facial structures, hair styles, and body types. Even though I was allowed to choose different color skin tones, the facial and body features were mostly of Caucasian decent. Therefore, people that I created looked like dark-skinned Caucasians. Furthermore, the Alice program only has two or three built-in African American characters in its gallery.

Another problem that Dr. Ellis and I worked on involved using Alice to allow users to place circles anywhere on the screen, then have Alice draw lines connecting to circles that the user clicked on. Next, a user would be able to click a color, select a circle, and then the circle would change to that color. Some limitations of Alice did not allow this project to go as planned. For instance, Alice cannot create objects at runtime, so it would not be possible to create a line that moves to connect one circle to another. Also, not being able to create objects at runtime limits the amount of circles that a user has to select and move around with because all of the circles would have to be created and placed in the program before it starts. With this new information in mind, Dr. Ellis and I decided that we would use Alice to develop a model of the Peterson graph and create a menu in Alice to allow a user to select a color of the circle. At the time of publication, this project was still ongoing.

3. SELECTED HOMEWORKS

In preparing for the class that Dr. Ellis will be teaching in the fall of 2006, Dr. Ellis developed a few homework problems for me to complete and return an estimate on the time that it took to complete these problems. The first homework problem created by Dr. Ellis instructed the creator to design an interactive program in which a person, who is holding a magnet, would point at metal objects to see what would happen. If the person is smaller than the object, he would move one meter closer to the object; however, if he was larger than the object, the object would move one meter closer to the person. Extra credit would be given for students exemplifying creativity beyond the requirements. In Figure 1 attached, the picture shows an Alice program that was developed to create the project.

When approaching the problem, I thought of using a skater holding a magnet with many of the objects laying on the ice. Movement would be easy because the skater could just slide on the ice when she was being pulled towards the object. This also allowed me to add creativity by doing tricks in between clicking on objects.

Another problem that we developed involved a horse race where a user was to click on a circle, denoting that he was betting on that particular horse. The horses would then race, using random intervals for movement. Extra credit would also be given for users demonstrating creativity. For example, the addition of a title screen would count as additional creativity. When first developing the program for this homework problem, I decided that I would use a random number to determine the winner of the race and specify the amounts that each horse would move to since the winner was already determined; however, after creating program, I learned that Alice does not seed the random number generator each time before it begins, and so unless I looped the program over and over, if a user restarted the game every time he wanted to play, the outcome would be exactly the same. After this, I decided that I would loop the game and so that the random number generator would keep running in through the sequence. Yet again, I ran into another setback; it was very hard to place the camera at the same initial position. And during the program, I had the camera start to move closer to the finish line as the game progresses. Since the initial camera angle was not exactly correct, by the time the program looped for the third or fourth time, the angles were very much off. Also, there was still a pattern in the winner of the program. Taking all of this into consideration, I decided that I would take a new approach to creating this horse racing problem. Instead of determining the winner, I decided that I would randomly generate the step lengths of the horses. Then using a while statement, I ran the program until the horses reached the finish line. If the horse that user selected reached the finish line first, then it was declared the winner. This also made for a more interesting race because the horses could jockey with each other and could create a come-from-behind victory. Figure 2 is an example of the horse racing problem.

4. DISCUSSION

By doing this study, we determined that it is very much so possible to teach the Alice programming language throughout the course of a semester. The drag and drop commands were very helpful in eliminating syntax errors and allowed me to specify to concentrate on the problems that I was working one. However, I noted many problems that I encountered in Alice.

As with any new development, there were setbacks that kept the program from performing at optimum level. Since Alice is not a text based editor, copying and pasting sections from one program to another is nearly impossible. Also, copy within the program can be difficult at times as well. For instance, if a user wanted to copy a group of code that contained variables and parameters from one world method to another method, Alice would flag errors unless the method in which the user is copying contains the exact same names of variables and parameters. Alice will not allow a user to copy over a section and then change the variable names to match the ones in the new method window. Another problem that I came across was with the camera. Sometimes, instead of telling the camera to turn backwards, move to or move away from objects, it would be better to specify by reference to where a user would like to have the camera move.

5. FUTURE OUTLOOKS

The current version of Alice that I have been working with is Alice 2.0. The developers of Alice are aware of many of the problems of the current version due to on-line message
Figure 1: Sample Solution to the Magnet Homework Problem

Figure 2: Horseracing Homework
boards and an error submittal database. Also, members of the Stage3 research team understand that the graphics of Alice are not up to standards with today’s generation. Therefore, the Stage3 research team has partnered with the makers of Electronic Arts the Sims © to produce the next version of Alice which will be called Alice 3.0. The new version of Alice will aim to fix the major of user problems and add fluidity and life-like motions to the characters of Alice. As of March 10, 2006, Stage3 has projected that Alice 3.0 will not be available for the next eighteen to twenty-four months.

6. REFERENCES

