

Motivating Programming:

using storytelling to make computer programming attractive to more middle school girls

Thesis Proposal
Caitlin Kelleher

Thesis Committee:
Randy Pausch (chair)
Jessica Hodgins
Alan Kay
Sara Kiesler

1. ABSTRACT

As of 2000, more than 50% of American homes had at least one computer and more than 40% of homes had an Internet connection. Despite the prevalence of computer users in our society, only a small, unrepresentative segment of our population can create new computer software. This is troubling for two reasons: 1) as the needs for technology continue to grow, we may be unable to fill computer-related jobs and 2) since people best understand their own needs, the technologies designed and created by an unrepresentative population may not adequately meet the needs of all the segments of our broader society.

To ensure both that we have a sufficient pool of qualified workers and that the new technologies we develop meet the needs of our entire population, it is important that we attract more members of underrepresented groups to Computer Science. Women comprise one of the largest underrepresented groups in Computer Science. I have chosen to focus on making learning to program attractive to middle school aged girls, who are old enough to be able to intellectually handle the concepts of programming, but have not already decided against pursuing math and science subjects, specifically computer science. By eighth grade, twice as many boys as girls are interested in pursuing science, engineering, or technology based careers.

Based on the attributes of middle school girls, I have chosen to focus on the application of storytelling. I believe that by presenting programming as a means to an end, through an application that supports storytelling, I can give middle school girls a positive experience with programming that will hopefully contribute to a more positive attitude towards computer science. To evaluate this approach, I will compare a storytelling-based introduction to programming with a more traditional approach to programming. Both will be based on the Alice system, a programming environment that allows novice programmers to create 3D virtual worlds. I will measure girls' attitudes before and after their introduction to programming and study what kinds of programs girls create and their experiences while using Alice.

2. INTRODUCTION

As of 2000, more than 50% of American homes had at least one computer and more than 40% of homes had an Internet connection (Newburger 2000). Despite the prevalence of computer users in our society, only a small, unrepresentative segment of our population can create new computer software. This is troubling for two reasons: 1) as the needs for technology continue to grow, we may be unable to fill computer-related jobs and 2) since people best understand their own needs, the technologies created by an unrepresentative population may not adequately meet the needs of all the segments of our broader society.

According to the US Department of Labor, 8 of the 10 fastest growing occupations between 2000 and 2010 will be computer related (Hecker 2001). The Information Technology Association of America reported that in 2002, after the dot com bubble had burst, companies were unable to fill 425,000 computer-related jobs due to lack of qualified applicants (ITAA 2001). Unless we can increase the numbers of students graduating with Computer Science degrees, the lack of qualified applicants for computer-related jobs may have serious consequences for US businesses.

However, even making our current work force larger (while maintaining the same level of gender and ethnic diversity) would not address all of the concerns about our current work force. The homogeneity of the current computer science community has created many technologies that were non-inclusive or worse, harmful towards certain groups within our population. Early speech recognition systems could not recognize women's voices (Margolis and Fisher 2002). Many still perform badly for people with different accents (Oviatt 2000). And, there are numerous other examples of technologies meant for everyone that were unintentionally designed for a subset of our population. As computer technologies become an increasingly integral part of everyday life, it is crucial that we make sure that our technologies serve everyone's needs, rather than just the needs of a particular group. Since people typically best understand their own needs, one of the easiest ways to make our technologies more inclusive is to have a more diverse group of people involved in designing and building the technologies we use.

Clearly, it is important not only to increase the numbers of people qualified to enter computer-related jobs, but also to make sure that those qualified people are a representative sample of our country's population. To get a larger, broader group of people to enter the field of computer science, we need to get a larger broader group of people to take the first steps towards pursuing a career in computer science. One of the main entry points for computer-related jobs is learning to

program. The ACM K-12 task force describes the relationship between computer science and programming in the following way:

While programming is a central activity to computer science, it is only a tool that provides a window into a much richer academic and professional field. That is, programming is to the study of computer science as literacy is to the study of literature (Tucker, Deek et al. 2002).

Learning to program provides students with the basic skills necessary to pursue computer science. If we can increase the numbers and broaden the spectrum of people who learn to program and who enjoy programming, we will likely help to create a larger more diverse group of people capable of succeeding in computer-related jobs.

However, even for those students who learn a little bit about programming, and ultimately choose not to pursue a computer-related field, there are benefits to having been exposed to programming. Few of today's students will be able to avoid working with computers in some capacity. Many students who choose not to pursue computer-related careers may find themselves working with computer scientists, programmers, and engineers in some capacity. Particularly for those students who end up working with computer professionals, a basic understanding of what programming is and how one writes a program will be helpful in preparing students to communicate and work productively with computer professionals.

Learning to program is also a valuable part of a general education for all students. This is reflected in the growing number of states that require some study of computing for high school graduation: 5 in 1996 (NCES 1998), and 11 in 2001 (NCES 2002). In addition to being a nice introduction to structured problem solving, programming also gives students experiences with complex systems. The world around us is filled with complex systems whose behavior depends on the behaviors and interactions of smaller parts within the system: cars, weather, and manufacturing plants, to name just a few. Yet our schools do little to prepare students to work with and attempt to understand the behaviors of complex systems. When your car breaks, it is helpful to be able to read a little bit about the main components of car engines and be able to eliminate possible problems based on your understanding of the behavior of your car. When we, as a country, make environmental policies that will impact neighboring states, countries, and the rest of our planet over many years, we would like our citizens to be able to recognize that seemingly simple actions like chopping down trees or drilling for oil, can have consequences in our air quality and food supplies. Programming provides children with some hands on experience dealing with complex systems that they create themselves. When their programs do not behave as

expected, children have to learn to isolate the problems and solve them. They learn to ask questions that will allow them to narrow the scope of a problem and they learn that a single malfunctioning piece within a program may cause other pieces of the program to break, as well.

While learning to program would be valuable for many students, it is also difficult for many students. Over the last 40 years, a number of researchers have attempted to make programming accessible to a broader range of people by reducing the difficulty of learning to program through simplifying syntax or preventing users from making syntax errors. More information on programming systems for novices can be found in the Related Work Section. While these improvements certainly make the process of learning to program less frustrating, they have not significantly changed the numbers or kinds of people who pursue computer science. Rather than focusing on making the process of learning to program less frustrating, I will focus on making it more attractive for a group of people who are not typically drawn to programming: middle school girls.

My focus on middle school girls is not arbitrary. Women are currently under-represented in Computer Science and related fields. By convincing even a small percentage of women to pursue Computer Science, we will help to create a larger and more diverse workforce. Middle school is the age at which many girls decide not to pursue math and science related disciplines, particularly computer science (AAUW 1996). By eighth grade, twice as many boys as girls are interested in pursuing science, engineering, or technology based careers (CAWMSET 2000).

However, as one of the main reasons for diversity is to involve people with differing perspectives on the future directions for technology in the process of designing new technologies, it is important that the group I choose to focus on also have a different perspective on where technology should be heading from those that are currently well represented in Computer Science. Fortunately, men and women of all ages tend to have different ideas about what kinds of technologies they would like to see in the future. Elementary and middle school girls and boys, when asked to design their “dream” technology, tend to describe very different things. Boys tend to fantasize about vehicles that can take them anywhere whereas girls tend to fantasize about objects that can help in everyday life (Brunner, Bennett et al. 1998). Researchers observed similar differences among adults. Men “tended to envision technology as extensions of their power over the physical universe” (Honey, Moeller et al. 1991). Women tended to envision

“technological instruments as people connectors, communicators, and collaboration devices” (Honey, Moeller et al. 1991). It is clear from these fantasies that girls and boys, and men and women will tend to push technology in very different ways, and that both viewpoints should be represented in the design of tomorrow’s technologies.

It seems clear that women tend to have different (and valuable) ideas about the directions for technology. So, how do we approach getting more women to enter computer and technology-based careers? The first step towards a computer-related career is often enrolling in a programming class. However, programming classes at both the high school and college level have traditionally failed to attract a significant number of female students (AAUW 1998). Perhaps the low enrollment of female students in programming classes and the comparatively higher enrollment of their male peers can be partially explained by a tendency of men and women to have differing interests in computers and computer science. Some researchers have found that many women are more interested in what they can do with computers than the computers themselves (Brunner, Bennett et al. 1998; AAUW 2000; Margolis and Fisher 2002).

In most computer science programs, students typically spend several years studying computer science before they reach a level at which they can use programming for applications that interest them. The first few years are spent learning how to use different programming constructs and data structures typically in pursuit of completion of programming tasks with very little practical value. A tool that displays the 17000th number in the Fibonacci sequence or simulates the line at a bank using a queue is not useful to most students. A student who is interested in what she can use programming to accomplish may quickly decide that this her introductory programming class is not very interesting and that perhaps programming is not used to solve interesting and relevant problems very often. Perhaps by presenting programming as a means to interesting ends, an approach that more closely matches researchers’ observations about women’s interests in computers and computer science, we may be more successful at attracting women into the field. There is some early evidence that an emphasis on interesting end-goals may help increase the enrollment of women in computer science courses: researchers at Georgia Tech have created an introductory programming course for non-CS majors that emphasizes the use of programming to manipulate digital media; the class is 2/3 female (Guzdial 2003). I believe that by creating a programming system that allows younger girls to view programming as a means to interesting ends, we may be able to attract more women to computing in high school, college, and later.

What constitutes an “interesting end” for a female student may vary considerably with age. To test the effects of a motivating end goal on girls’ interest in learning to program, we will need to focus on a single age group. Studies have shown that middle school is a critical age for girls; it is during the middle school years that many girls will decide whether or not to seriously pursue the study of math and science (AAUW 1996). By late high school many girls have already opted out of the math and science classes that would make it possible for them to pursue a mathematical or scientific major (AAUW 1998). To have the most impact, we need to give girls a positive exposure to programming no later than middle school. In my thesis work, I will focus on building a programming system that will enable middle school girls to have a positive experience with programming before they have decided against further pursuing math and science.

2.1 Middle School Girls

To successfully design a programming system for middle school girls, it is important to understand this age group. In this section, I will attempt to provide some insight into what a middle school girl is *like*. Most of the information reported here is based on research that included many girls, so any single girl may not exhibit all or perhaps even any of the characteristics of the broader group. Most girls will have at least some of the characteristics of the broader group.

2.1.1 Gender Expectations

Most girls hit puberty at twelve; middle school children are typically between twelve and fourteen (Collins and Kuczaj 1991). In our culture, as children start to exhibit the physical changes of puberty, many people begin to expect both more adult behavior and more gender-appropriate behavior from then (Collins and Kuczaj 1991). Since there is still some disagreement on how girls and women should behave, girls at this age often receive very confusing messages from society (AAUW 1996). In their report *Girls in the Middle*, the American Association of University Women described the expectations for adolescent girls this way:

Adolescent girls are to be sexy and flirtatious but at the same time remain “good girls.” They are to fend off aggressive male attention while simultaneously meeting teachers’ expectations of non-aggressive behavior. Females are to put domestic life first at the same time that they prepare for financial independence (AAUW 1996).

As adolescents start working towards determining how to balance the sometimes conflicting expectations for young men and women, they tend become much less accepting of gender transgressions (behavior associated with the opposite gender) in their peers than they were just a

couple of years before (Collins and Kuczaj 1991). One study revealed that adolescents are hesitant to partake in activities that they perceive as belonging to the opposite gender because they believe that it may cause or at least be indicative of a problem with sexual identity (Collins and Kuczaj 1991).

2.1.2 Academics

During the middle school years, many girls experience a drop in self-esteem and confidence in academics, particularly in math and science (AAUW 1996). In the third grade, approximately the same number of boys and girls believe that they are good at math (64% of girls and 66% of boys) (AAUW 1992). By seventh grade, 57% of girls and 64% of boys believe they are good at math. That number of girls who believe they are good at math drops to 48% by the end of high school (AAUW 1996). This confidence drop typically *precedes* a drop in academic performance (AAUW 1992). Girls' experiences with science are similar. From elementary school through high school, girls are less likely to do science based activities than boys (AAUW 1996). While girls express interest in science in the elementary years, they have increasingly negative views of science, science classes, and science-based careers as they progress through middle and high school. While the exact reasons for the self-esteem drop and later performance drop in math and decreasing interest in science are unclear, there are a couple of known factors that may contribute. Girls tend to attribute their own struggles and failures to a lack of ability, which may encourage them not to put as much effort into their schoolwork (AAUW 1992). Boys, on the other hand, tend to attribute failures to insufficient hard work or, bad luck (AAUW 1992). Additionally, excelling in school may lead to social isolation for girls, as their peers begin to lose academic confidence and downplay the importance of academics (AAUW 1996).

2.1.3 Identity Formation

For girls and for boys of this age, the process of forming an identity is a fundamental activity of the middle school period (Stone and Church 1984). Psychologists define identity as an "internal, self-constructed dynamic organization of drives, abilities, beliefs and individual history" (Marcia 1980). In other words, identity is a person's own ideas about what she stands for, what she is good at, what she enjoys doing, and how her past helped to shape who she is today. Identity is dynamic, so it can change to adapt to new life experiences, new friends, and new found abilities or failures (Stone and Church 1984).

Psychologists believe that identity formation is a process that must include two elements: a crisis and a commitment (Stone and Church 1984). During the crisis, children become aware that they

are expected to fill or begin to anticipate adult roles and start to feel pressure to integrate information and emotions about themselves and their experiences (Marcia 1980). The crisis is resolved when a child commits to a role or set of roles that he or she is currently filling or working towards filling as an adult and a set of beliefs about themselves and the world they inhabit (Marcia 1980). Children who do not experience the crisis often accept the roles that parents, teachers, peers, or other significant figures in their lives want for them (Marcia 1980).

To resolve their identity crises, middle school aged children typically experiment with a wide variety of roles in their interactions with peers, teachers, parents, etc (Frankel 2002). Before settling on a role or set of roles, middle school aged children may change roles and personae rapidly (Frankel 2002) and may take on different roles when interacting with different people (AAUW 1996). A girl of this age may try to appear daring and independent with her friends, obedient and responsible with her teachers, and childish and playful with her parents as she tries to sort out what she stands for, what her strengths are, and what she enjoys.

2.1.4 Middle School Girls and Computers

Girls are now using computers in large numbers. As of 2000, 9 out of 10 children in the US have access to a computer in school or at home (Newburger 2000). According to data from the 2000 census, over 66% of households with children between 6 and 17 have at least one computer (Newburger 2000). Among teens 13 to 17 years of age, 73% of girls and 70% of boys use the Internet, although their usage patterns tend to differ (Grunwald 2003). Girls tend to do more communication-based activities: 68% of girls report using email “very often” or “pretty often”, as compared to 50% of boys (Roper 1999); 56% of girls report using instant messaging “very often” or “pretty often”, as compared to 48% of girls. Boys are more likely to report that they play games (50% of boys vs 43% of girls) or gather information about sports (40% of boys vs 15% of girls) “very often” or “pretty often” (Roper 1999). Still, girls are unlikely to move from using computer software to signing up for a computer programming class. Potential reasons for this include the perception of computer science as a male domain and the fact that traditional presentations of computer science are not interesting.

When asked to draw someone who is good with computers (a computer whiz) the computer whiz is male in 71% of responses, female 18% of responses, and indeterminate in the remaining 11% (Castell and Bryson 1998). Girls view the heavily technical culture, with its emphasis on clock

speeds, megabytes and other performance metrics as a fairly uninteresting domain and one that is most appropriate for boys (AAUW 2000). Since adolescents are less tolerant of cross-gender activities than kids of most other ages, the perception of computer science as a boys' activity can itself be an inhibitor, even for girls who are interested (Collins and Kuczaj 1991).

Further, middle-school aged girls, like women of all ages, tend to approach computers more as a tool than a toy. They tend to be more interested in the potential applications for computers than they are in computers as artifacts. Traditional presentations of computer science do not allow students to think about relevant applications until they have studied computers for a few years. Many girls have an "We [girls] can, but I don't want to" attitude towards programming and computer science classes (AAUW 2000). While this is perhaps partially a reaction to the difficulty in getting started learning to program, it is almost certainly also a reaction to the beginnings of most programming classes, in the words of one girl "it takes so much just to do something simple with a computer" (AAUW 2000).

2.1.5 Properties of a Programming System for Middle School Girls

In general, girls seem to turn away from programming more for lack of interest than lack of confidence in their abilities, although a lack of confidence probably also contributes (AAUW 2000). During middle school, girls are occupied primarily with the task of creating an identity and highly value their relationships with peers (Collins and Kuczaj 1991). Based on what is known about middle school girls, I have developed a preliminary list of attributes of a programming system for this group.

1. The system should provide and focus on a motivating end-goal that allows girls to build something they are personally interested and invested in early in the experience. To a middle school girl, programming is likely to be little more than a means towards a goal she finds motivating.
2. The programming system should be based around an activity that is not normally considered something "for boys." Middle school children, both boys and girls, are very reluctant to participate in activities that obviously belong to the other gender. This reluctance is more present in middle school than at almost any other age.
3. The system should allow girls to work together in pursuit of a common goal. Girls place a lot of importance on their relationships with their peers.
4. The system should allow girls to create something that a non-programming friend can appreciate. Positive feedback from a peer is extremely valuable.

5. The end-goal should be related to the task of identity formation.

3. APPROACH

3.1 *Alice System*

To leverage previous work on simplifying the process of learning to program, I will base my programming system for middle school girls on an existing programming system for novice programmers called Alice (www.alice.org 2003). Alice allows novice programmers to create virtual worlds, typically short animated movies or games. Students create programs by dragging and dropping rounded-rectangular tiles that represent commands. Figure 1 shows the Alice interface and describes each of the areas of the interface. In the figure, the user has just dropped a “move” tile, and is in the process of selecting the direction and distance for the IceSkater to move.

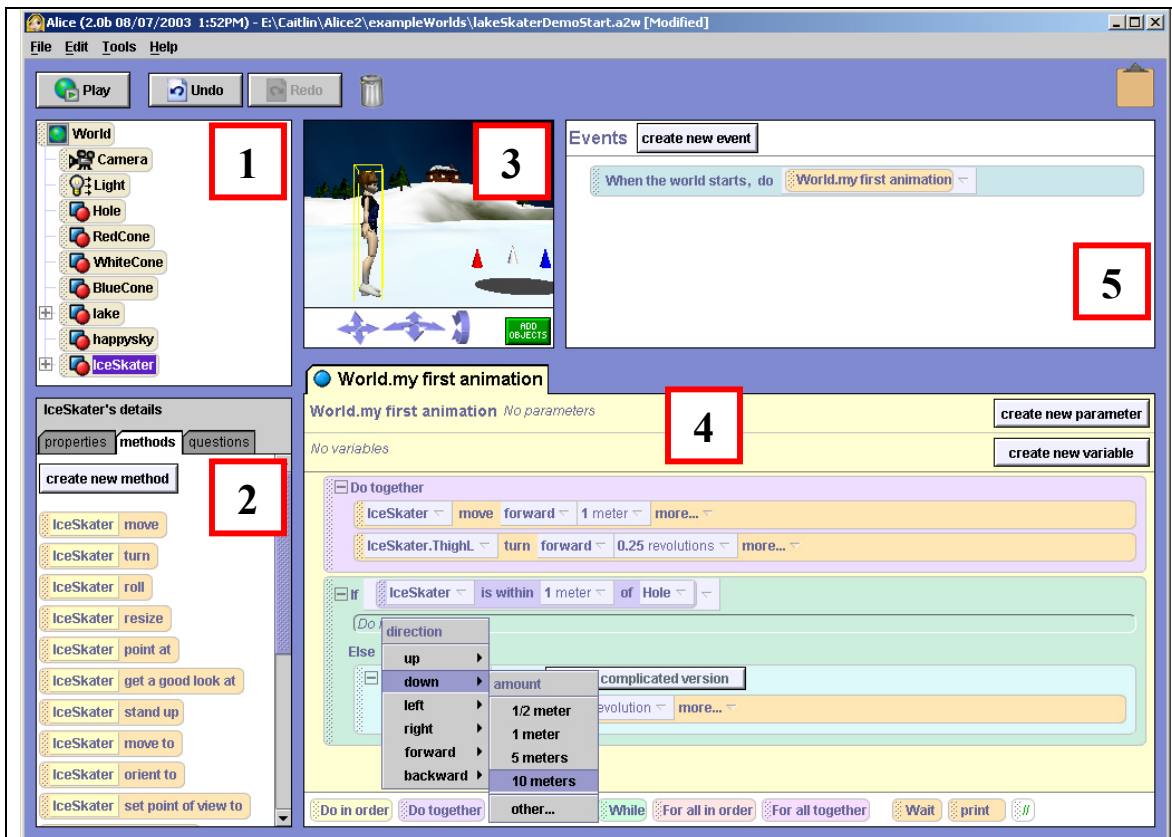


Figure 1: A screen shot of the Alice system; each interface area is labeled. Descriptions of each area are below.

- 1: The object tree displays a list of objects in the current Alice world and allows students to select objects.
- 2: The details area displays the properties, methods, and questions for the currently selected object.
- 3: The scene editor displays the 3D world and allows students to place objects and navigate through the virtual world
- 4: Students can build animations and questions by dragging program tiles from the details area.
- 5: Students use behaviors to associate methods with events such as mouse clicks.

Because users construct programs in Alice by dragging and dropping program tiles and then selecting parameters from a list of valid choices, they cannot make syntax errors. By preventing users from making syntax errors, Alice allows users to gain experience with and understanding of the logic and control structures used in programming, independently of learning to type syntactically correct program statements. While other systems have prevented users from making syntax errors (Kay; Finzer and Gould 1984; Repenning 1993; Lionet and Lamoureux 1994; Miller, Pane et al. 1994; Gindling, Ioannidou et al. 1995; Logo Computer Systems 1995; Begel 1996; Kahn 1996; Smith, Cypher et al. 1997; Hartmann, Nievergelt et al. 2001; Logotron 2002), most limit users to a small subset of the structures typically found in most common programming languages. Alice allows users to gain experience with all of the standard programming constructs taught in introductory programming classes, such as loops, if-statements, recursion, variables, and parameters.

Alice is currently being used to introduce programming to at-risk college students. These students lack programming experience and do not have sufficient grounding in mathematics to enroll in Calculus but want to major in Computer Science. Typically, at-risk students perform poorly in their first Computer Science course (CS1) and only rarely enroll in the second Computer Science course (Cooper, Dann et al. 2003). Rather than having at-risk students begin with CS1, Cooper, Dann, et al are studying the effects of having at-risk students take an Alice programming class either prior to or concurrent with CS1 (Cooper, Dann et al. 2003). Early results indicate that students who take an Alice programming class perform a letter grade better and are four times as likely to continue to CS2 than at-risk students who enroll only in CS1 (Cooper, Dann et al. 2003). At-risk students who take an Alice class earn grades comparable to those of students with stronger mathematical background and programming experience and are more likely to continue to CS2 than their more experienced peers (Cooper, Dann et al. 2003). These early successes at the college level, particularly among female and minority students, make Alice a strong starting point for my work.

Before committing to base my work on Alice, it is important to establish that girls will not reject Alice as being a domain for boys. Over the last year and a half, I have worked with more than 70 middle school girls in Alice workshops. In my experience, girls do not see Alice as a programming environment, they typically view Alice as a program for making animated movies, an activity that they do not strongly associate with either gender.

3.2 *Storytelling as a Motivating End-Goal*

To introduce programming as a means to an end, it is important to choose an end goal that middle school girls will find motivating. Storytelling has a number of properties that make it a promising application for middle school girls. Storytelling is a form of self-expression and a method for allowing girls to try out different roles in scenarios that are most relevant to them. Role-playing is one of the main strategies that adolescents use in developing their own identities. In early experiences observing middle school aged children (mostly, but not exclusively girls) attempting to create stories in Alice, I have seen stories with themes ranging from how to relate to boys, to what to do if your boyfriend or girlfriend is cheating on you, to the difficulties of dealing a seemingly endless stream of criticisms from adult authority figures. While some girls may eventually want to use Alice for other purposes such as creating simulations or games,

storytelling as an introduction is likely to have wide appeal; there are few people that do not appreciate stories in some form.

Girls' use of role-playing in identity formation might suggest that I should concentrate my efforts on computer-based role-playing games, either the dungeons and dragons style games or those that are more character and story centric. I am not aware of any role-playing games that both allow users to program and provide the same level of support for novice programmers that the Alice system provides.

Since I intend to base my programming system for middle school girls on the Alice system, it is important to choose an end-goal that Alice can reasonably support. Alice is designed to allow users to create interactive virtual worlds; animated stories can be thought of non-interactive virtual worlds and are a natural subset of Alice's design space.

3.3 Thesis Statement

An introduction to programming for middle school girls that presents programming as a means to a motivating end will be more successful at getting girls interested in programming than a traditional approach that focuses on teaching programming as an end in itself. To successfully introduce programming as a means to an end, it will be necessary to create a programming system that provides sufficient support for the end goal to allow girls to focus primarily on that end goal. Storytelling is likely to be a motivating end goal for a broad spectrum of middle school girls.

A programming system for creating stories should have the following properties:

- 1) The system should clearly communicate to girls that it can be used for the purpose of telling stories.*
- 2) The system should provide support to help girls generate story ideas and steer them towards kinds of stories with which they can be successful early on.*
- 3) The system should provide commands at a sufficiently high level to allow girls to focus primarily on creating stories rather than requiring them to focus on low-level details. .*

Girls who are introduced to programming as a means to a motivating end, such as storytelling will show more evidence of engagement than girls introduced to programming as an end in and of itself. I will be able to find quantifiable behavioral differences between the two groups, such as number of lines of code or time spent working on programs.

In the remainder of this section, I will further discuss each of my sub-theses, prior work I have done towards them, and my research plans. As much as possible, I plan to quickly prototype solutions to observed problems, test them with real users, and iterate.

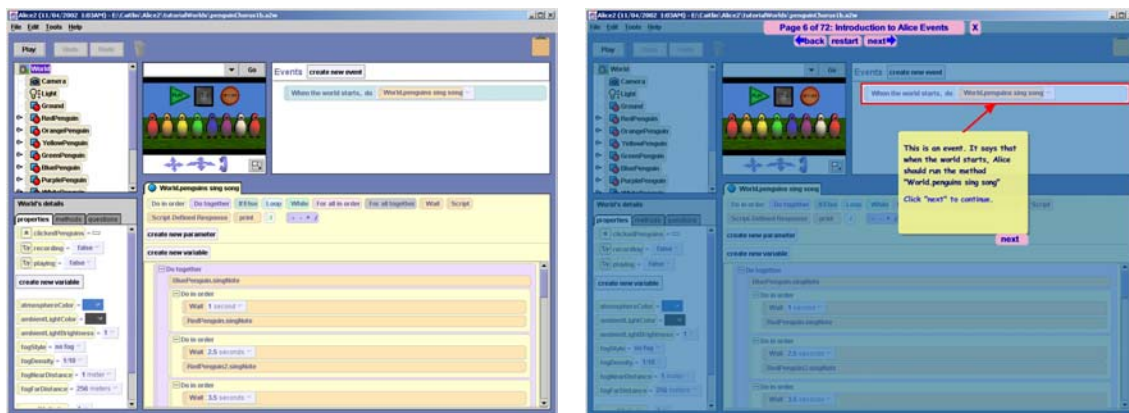
3.3.1 The programming system should clearly communicate to girls that it can be used for the purpose of telling stories.

For most girls, their first exposure to using Alice will likely come through the online tutorial provided with the Alice system. After completing the tutorial, girls should be ready and interested in trying their hand at creating a story in Alice. Traditionally, the Alice tutorial has attempted to provide users with an introduction to the functionality of the system using examples that are as simple as possible to minimize the number of potential errors that a new user might make. While successful in showing new users how to do basic tasks within Alice, it is less successful in showing people what Alice is for. If, after completing the tutorial, girls don't have a more compelling answer to the question "what can I do with this program?" than "move objects around in a scene," they will be unlikely to continue. A tutorial must teach users how to use the system, but it must also convince them that the system will allow them to do something they find interesting or worthwhile.

3.3.1.1 PRIOR WORK

To get and keep girls interested in using Alice, I have redesigned the tutorial using a series of example projects that girls may be interested in: teaching an ice skater a new routine and writing a story about a bunny rudely awoken by a cell phone. While the new tutorial examples are more interesting and better tell the story of what Alice is for, they also involve creating worlds that are more complex. This tends to exacerbate common problems users have with tutorials: difficulties finding the correct component in the interface, accidentally skipping steps, and performing actions not intended by the author of the tutorial (Knabe 1995). More complex worlds require more components to be displayed in the Alice interface, often requiring users to spend more time searching before they locate the correct component. Complex worlds also contain more overlapping screen components, allowing users to potentially obscure the component they are searching for. Skipping steps or performing actions not intended by the author of the tutorial can easily put the interface into a state that a novice user cannot easily recover from. To make it feasible for novice users to complete tutorials involving more complex worlds, it is necessary to find a new way to present tutorials that removes as much of the unnecessary potential for error as possible.

The only tutorial in which no user will ever make a mistake is one in which no user ever performs an action. While I could present a video-style tutorial for Alice, users benefit significantly from actively working through the tutorial themselves (Muir 2001). Therefore, I want to maintain the property that the user must perform the actions dictated by the tutorial. But, I also want to make it take less time for users to find the component they are supposed to interact with in the current step and reduce the chance that users will skip a step or perform an incorrect action. Effectively, I would like to give users an experience similar to that provided by a person who knows a piece of software walking the users through a new process, pointing out the components they need to interact with, telling them what actions to take, explaining those actions, showing them the results of your actions, and preventing them from clicking on components that are not necessary to complete the current step. To do this, I introduce a new interaction technique, called Stencils (Kelleher 2002), in which I place an overlay on top of the Alice interface that captures mouse and keyboard events. Stencils can contain three basic types of objects: notes, holes and frames. Notes are sticky-style notes that provide either general information or information relating to a specific interface component. Frames draw the users' attention to a particular component within the interface by drawing a red rectangle around it. Holes provide sections over specific components that allow users to interact with only specific components. See Figure 2 below. A tutorial consists of a set of stencils that guide a user through a sequence of steps.



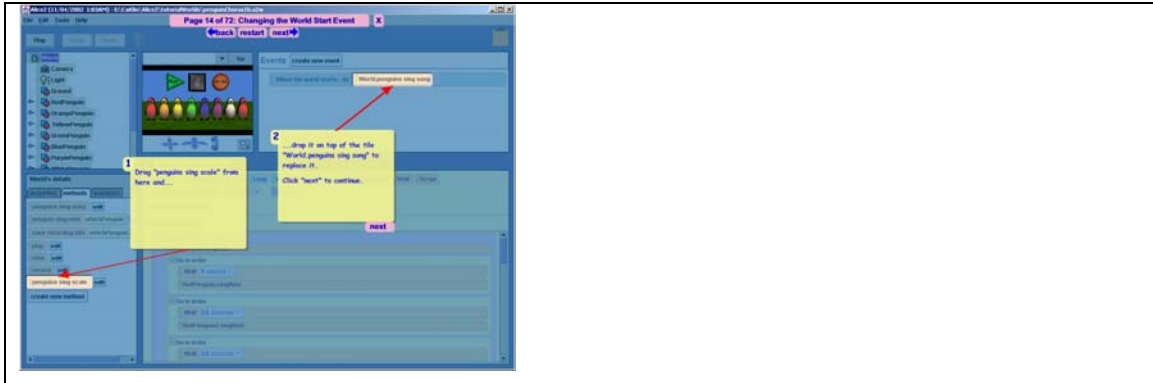


Figure 2: The Alice interface with Stencils

Top left: The Alice interface without a stencil

Top right: The Alice interface with a blue stencil containing a frame overlaid. The stencil is highlighting changes the user has made to their Alice world.

Bottom left: The Alice interface covered by a blue stencil with two holes. The stencil is directing the user to drag a method from the hole in the lower left hand corner into the behavior area in the upper right hand corner of the Alice interface.

Early testing of the new tutorial seems to indicate that middle school aged girls are more interested in Alice after completing the new tutorial. After completing the old tutorial, it was rare to find a girl who was interested in continuing to use Alice. With a pilot group of 25 middle and early high-school aged girls using the new tutorial, 20 indicated that they were interested in continuing to use Alice. While these results are anecdotal, they are nevertheless encouraging.

I have developed Stencils to make it easier to guide middle school aged users through complex interfaces within Alice. However, this technique can be employed in creating tutorials and help systems in other graphical user interfaces for a broad range of people. I have conducted preliminary user tests with more than 50 users, of both genders and ranging in age from 6 to 70 years. Based on these tests, Stencils appears to be usable by a broad group of users but is particularly helpful for users who are afraid that they may make a mistake that they cannot recover from.

3.3.1.2 RESEARCH PLAN

Developing and testing a preliminary version of a storytelling-based tutorial using the Stencils interaction technique with middle school girls has given me confidence that the approach of introducing programming through storytelling has promise for getting middle school girls interested in learning to program.

The remaining work on Stencils lies in two areas: improving Stencils and leveraging Stencils to give users access to more advanced information when they are ready for it. It is still possible for users to make errors using Stencil-based tutorials in some cases. For example, users may select the wrong item from a menu or call animations in the wrong order inside a method. To handle these cases, I will add a state-checking capability to verify that users have performed the intended action before allowing them moving to the next step. In a situation in which users can perform any action available in the interface, state checking can be extremely expensive. Fortunately, using Stencils this problem is simplified by the fact that only certain actions are possible within the context of the tutorial, so it is necessary to check for a relatively small number of errors. I will also investigate the possibility of extending Stencils to create a help system that can help students learn a broader range of programming concepts and Alice features.

3.3.2 The programming system should provide support to help girls generate story ideas and steer them towards kinds of stories with which they can be successful early on.

The Alice system provides a gallery of 3D objects (see Figure 3 below) for use in Alice worlds that consists of over 1000 models of characters, scenery and objects. Every Alice object knows how to move, turn, and perform other basic animations. Some Alice objects also know how to perform object-specific animations. For example, a ballerina might have animations that represent common ballet movements; a fireman probably would not. The gallery is grouped into a variety of categories, typically either based on a shared attribute (i.e. all vehicles are grouped together) or a theme (i.e. all Alice objects relating to Ancient Egypt are grouped together). When middle school aged users start to create their own animated story in Alice, they often browse through the gallery searching for ideas. Many users seem to gravitate towards themed groups of objects.



Figure 3: A small sample from the over 1000 objects currently in the Alice Gallery

While girls typically seem excited about creating their own animated movie, they often have difficulty finding a compelling story to tell with Alice, particularly when they have chosen a set of characters and objects that do not naturally lead to story. One middle school aged girl demonstrated this difficulty well: she carefully added all the rides and props from the amusement park to her world and a man to be in the amusement park. But, beyond having the man ride a few rides and say they were fun, she couldn't think of anything for the man to do and got bored within 10 or 15 minutes.

Many middle school girls use the content in the gallery to help form a story idea. The current large gallery seems to steer girls towards creating large, detailed environments that often do not suggest potential stories. I believe that is possible to create and organize gallery content in a way that will help users get started telling stories.

3.3.2.1 PRIOR WORK

To begin exploring how to support storytelling through content in the Alice gallery, I led a class of 13 undergraduate students in which teams of three students created Alice StoryKits, themed collections of Alice objects designed to support story. Each group produced one kit every 2-3 weeks. The group deadlines were spread out such that we had two new kits to be tested almost every week during the semester. Each Friday we invited 10 middle school aged children (7 girls and 3 boys) to come and build stories using the StoryKits produced by the undergraduates while we observed. In observing the middle school students, we focused on determining the techniques they used in searching for stories and the kinds of content that helped the middle school students

find ideas for stories. Based on these observations, we were able to iterate four times over the course of the semester and develop some basic guidelines for producing successful StoryKits.




	<p>The Robots StoryKit contains 5 robot characters: two girls and two boys (for couple stories) and a wacky professor (for Frankenstein-style stories). The students who created this kit chose a robot-theme partially to match Alice's somewhat rough animation-capabilities.</p>
	<p>The Robots StoryKit comes with a factory setting: a building and a couple of factory machines.</p>
	<p>Both the characters and scenery objects come with several animations: the couple-robots have flirting-themed animations; the professor can explode; the factory machines can blink and turn on and off.</p>

Figure 4: An Example Storykit

Preliminary Design Guidelines for StoryKits

- Environmental
 - Create stage-like sets in which users can see all the elements of the world rather than large, open scenery.
 - Use objects in the scenery that suggest events or tension. For example, a well-placed “No Trespassing” sign can often suggest a starting point for a story.
 - Initial staging may also be used to suggest conflict.
- Story
 - Characters alone are not enough – users need some help figuring out what might happen.
 - Cliché stories and stereotyped characters are safe bets. At this age, they are understood and enjoyed.
- Character
 - Middle School Children enjoy distinct, strong characters. Even the tiny kid on the playground that everyone picks on should have some special ability or power.
 - Characters should have specific and obvious roles.
 - Middle School Children get most creative in their use of dialogue – set up stories that encourage characters to converse.
 - Middle School Children enjoy interactions between characters, both positive and negative. Where possible, enable characters to hug each other, make eyes at one another, or kick each other. Girls seemed to enjoy the use of cartoon violence as much as boys, but their stories include reasons for the violence.
- Animations
 - Use expected animations; characters should be able to walk, sit in chairs, etc.
 - Enhance characters with unexpected animations, but use ones for which kids can find lots of motivations. A big strong ogre might cry or suck his thumb, but having him be able to do strange tricks with his body parts (spin his horns or his head, for example) is less useful.
 - Include animations that imply conflict.

3.3.2.2 RESEARCH PLAN

I plan to continue refining the Alice StoryKit design guidelines. Using StoryKits, most pairs of middle school aged girls are typically able to create a story that they are pleased within one to two hours. Without StoryKits, most pairs of girls commonly gave up after 30-45 minutes of trying to

build an animated story in Alice, either because they could not find a story idea or because they could not successfully realize their story idea. The four rounds of producing and testing Alice StoryKits have provided some insights into the kinds of content that help middle school students get started telling stories, but the guidelines are still vague. I will continue to refine the StoryKit guidelines. In particular, I will try to identify the most important factors in producing a successful StoryKit and more concretely identify how to use those factors in the creation of successful StoryKits. While it is necessary to have a sufficient number StoryKits to allow girls to feel they have sufficient choices for stories, too many StoryKits may be overwhelming. I will also try to determine the optimal number of StoryKits.

As I continue observing middle school girls creating stories in Alice, I will look for other ways that the system can encourage storytelling. For example, when we began creating StoryKits, we expected that middle school girls would “audition” characters for their stories by adding them to a world, trying out their animations, and then either keeping them or deleting them, as appropriate. Instead, girls seem to focus mainly on the visual information available about characters from the gallery. As a consequence, they often overlook characters that have interesting animations. One way that Alice can help to encourage storytelling is by helping girls find content that is interesting to them quickly. Through user testing, I will identify other story-based tasks the Alice system can better support and types of information the system should reveal to girls to help them create stories. Based on the results of user testing, I will incorporate additional storytelling supports into the Alice system.

3.3.3 The system should provide commands at a sufficiently high level to allow girls to focus primarily on creating stories rather than requiring them to focus on low-level details.

Even when middle school girls have a story idea they want to create in Alice, it is often difficult for girls to realize their stories. I observed a test group of nine girls (six in pairs and three alone) who completed the tutorial and then proceeded to attempt to create their own story. After adding a few objects into their worlds, the girls typically described to each other an opening sequence of a story. One pair of girls wanted to begin with a boy coming into view on his skateboard and seeing a girl on her skateboard doing fancy tricks. They quickly discovered that this type of scene is difficult to create; they needed to get the boy to walk to his skateboard, get on, and have them move together. This requires they create a walk animation, teach the boy to get onto the skateboard, teach the skateboard to move appropriately, etc, and all for the first half-sentence of

their story. Particularly when your end goal is a story, primitive animations, like move and turn, can be very frustrating. Rather than concentrating on stories, users often have to spend several hours building a suite of basic animations such as walk or wave hello before they can begin really working on the story that they envisioned. This can be frustrating, particularly for middle school girls who are likely to be focused on a story that they want to tell. If they cannot achieve significant progress in a short amount of time, most will give up.

Several studies have shown that people write approximately the same number of lines of code in a given amount of time, regardless of the language that they are using (Gray). I believe that the further a girl can get towards creating a story in her first hour after the tutorial, the more likely she will be to continue working on that story and to continue working with Alice. To enable kids to develop something they are interested in more quickly, it will be necessary to provide a well-chosen set of higher-level animations that will allow users to get started telling stories more quickly.

3.3.3.1 PRIOR WORK

While providing higher-level animations and helping users get started telling stories are distinct problems, it would be difficult to address them completely separately. Until users have a story idea they can pursue, it is difficult to determine what the requirements are for higher-level animations. And, while they are struggling with difficulties in realizing their story ideas, it can be difficult to determine if the story idea supports are sufficient. In addition to developing guidelines for storykits, the Alice StoryKits class observed several tasks that middle school aged children struggle with in attempting to create stories in Alice and included some prototype solutions to those problems in their StoryKits.

Over the course of the semester, the Alice StoryKits team observed 3 main areas that appeared to cause the most frustration for our middle school test users: users had no good methods for communicating backstories; users often had trouble positioning the camera to get a good view of the action in their story; and, they often had difficulty getting their characters in good positions relative to each other. I describe each of these three problems in more detail below.

3.3.3.1.1 Communicating Backstory

Users sometimes come up with elaborate backstories and want to present them as dream sequences or separate scenes in the past. More often, they just describe what happened in a sentence or two. But, for the past seven years, Alice has only provided animations that allow

Alice objects to move or change the appearance. As a first attempt to help support users communicating their stories, I added “say” and “think” animations that display cartoon speech or thought bubbles for Alice objects. The ability to have characters talk or think goes a long way in helping middle school girls communicate their stories, but users often want to communicate the emotions or histories of their characters. As I first step, I will add an animation that allows users to create textual subtitles to provide more information about their stories and characters.

3.3.3.1.2 Blocking characters

In story-based worlds, users often need to move characters to positions relative to each other. For example, a user may want to start a scene within her story by moving one character so that she is standing in front of another character so that the two characters can talk. In drama, the placement and movement of actors throughout a play is called *blocking*.

Currently, there are three animations within Alice that users tend to try to use in attempting to block their characters: move, move to, and move toward. Move allows a given character to move some distance in an object centric direction (forward, backward, left, right, up or down). “Move to” moves a character to a particular location in space, often the center of another character. Move toward moves a given character, without turning the character to face the direction that he or she is going, along a line between the character and what he or she is trying to move toward. Move and move toward require users to iteratively guess how far their characters need to move. And, if they ever move the characters in the opening scenes, all their hard work is lost. “Move to” almost never has a useful function for novice users. Using these three animations, users are often frustrated because tasks that seem simple, like moving a character to stand in front of or beside another character, can prove quite difficult (Conway 1997).

3.3.3.1.3 Controlling the Camera

Unlike the first two problems, which have clear first steps, it is not immediately obvious how users will want to control the camera. While other researchers have approached the problem of camera control, most have focused on the needs of expert animators (Gleicher and Witkin 1992; Drucker and Zeltzer 1995) or on automatic camera controls end-users of virtual environments (He, Cohen et al. 1996; Tomlinson, Blumberg et al. 2000). Many users have specific criteria for what should or should not be in a given scene and would likely be frustrated by a fully automatic solution. Instead, I will need to further investigate how middle school girls want to move the camera and provide support for the commonly needed camera motions.

3.3.3.2 RESEARCH PLAN

While it is not feasible to provide full solutions to all of the problems discussed here within the context of this work, it is important to understand the range of problems that users may encounter. I will select a subset of these problems through iterative testing, working towards addressing the most crippling problems observed during a given round of testing in the next round of design.

3.3.3.2.1 *Communicating Backstory*

As a first step, I will add an Alice animation, like say or think, which allows users to easily add narration to their worlds (i.e. *Alice.narrate*("Long, long ago, in a land far away..."). Through working with middle school girls, I will determine where *narrate* is helpful and what other kinds of information girls want to be able to express in their stories, either textually or visually.

There is a wide variety of information that girls may want to communicate as part of their story, elements such as a character's internal state or history, a past event, an event occurring in another location, the passage of time, a dream sequence or a fantasy. While many of these can be at least hinted at through narration, some users have wanted to show these elements instead of simply describing them. To support communicating more complex story sequences, we will examine how films, television, and cartoons show things like past events, dream sequences, and fantasies and develop supports (in the form of Alice objects and/or animations) that enable users to easily establish a context for a non-sequential element in a story. Also, many of these will require the user to set up and use multiple scenes. While this is currently possible in Alice, it is not something a novice is likely to find on his or her own. We will conduct user studies to determine the best way to set up multiple scenes within Alice and provide this support.

3.3.3.2.2 *Blocking characters*

As a first pass in making the process of blocking characters more tractable for novices, we will change the move to animation to take an object and allow the user to specify a distance and direction from that object. Since most of the time characters will want to face each other at a conversational distance, we will choose the defaults for the optional parameters *howFar* and *whichDirection* so that this will be the default behavior.

SALLY MOVE TO JACKIE HOWFAR= 1 METER WHICHDIRECTION= FRONT

As previously discussed, move and turn are at the wrong level for easily creating stories. Middle school aged girls telling stories in Alice often need to be able to position characters relative to each

other. As a first step, I will make the “move to” animation support positioning characters relative to each other.

Blocking is one of the first difficulties that girls often hit when creating stories in Alice, but there are other kinds of actions they are likely to want to do that Alice does not currently support gracefully. I have broken the kinds of animations that I think may be necessary into three types: character animations that do not depend on anything else within the world, character-character animations in which two characters interact with each other, and character-object interactions in which a character is performing some action involving an object. I will focus on identifying the most commonly useful character animations for storytelling and find ways to supply those animations for Alice objects and characters. Based on observing approximately 70 middle school girls attempting to create stories over the last year, I have found that middle school aged children seem to strive more for the ability to communicate their story to viewers than absolutely realistic animation. As long as it is clear that a given character is trying to pick up an object, it does not matter whether or not his fingers close appropriately around the handle.

3.3.3.2.3 Character Animations

There seem to be only a few universal individual character animations: locomotion and emotion.

In developing Storykits, we have prototyped many locomotion animations for many different characters. Again, users want someone to be able to say, “Oh, he’s walking” but do not tend to get concerned about the angles at which his feet hit the ground or how graceful the first and last steps are. As a consequence, we should be able to provide a set of simple locomotion animations. There are two potential approaches to providing locomotion: 1) providing a method for generating people with a canonical hierarchy of body parts or 2) creating locomotion animations that can be applied to a wide variety of body part hierarchies. The current gallery includes over one hundred humanoid characters. If possible, I would like to leverage these models. The main complicating factor will be that there are a wide variety of object hierarchies, even among the human characters, within the gallery; it may prove very difficult to create locomotion animations that can work even for a majority of these models.

To create a set of emotionally based animations, we will begin with the work of Delsarte, an 18th century dramatist who codified emotional expressions for use in acting (Stebbins 1902). While

his approach to acting is not widely used within the drama community because it tends to result in fairly “iconic” characters, this approach may work well within an animation context.

Character-Character Interaction Animations

Users often want their characters to interact with other characters. Examples of interaction include touching (e.g. kiss or slap), touching and following a motion (e.g. shake hands), complex contacts (e.g. hug or place arm around shoulder), and relative positioning (e.g. walking to stand in front of another character). Relative positioning is the easiest, and most common kind of character-character interaction. Touching, both with and without following, would probably require the use of some inverse kinematics.

Character-Object Interaction Animations

Interactions with objects are somewhat similar to interactions with characters. Users would like to be able to pick up objects, touch objects (either with their hands or with other body parts), or follow a path with their hand (e.g. opening a door).

Given the potentially wide usage for inverse-kinematics based end-effector animations, I will likely explore these as the next step in developing animations at a more appropriate level for storytelling.

3.3.3.2.4 Controlling the Camera

To gain a better understanding of the requirements for camera motion, I will collect storyboards and descriptions of where the camera is located from users and study the ways in which they draw their scenes. Additionally, I will look at the common types of camera shots used in movies, television, and cartoons, as these are the media that girls often try to emulate. Faculty and students at CMU’s Entertainment Technology Center will be a valuable resource to me in studying the use of cameras in other media. Using both the storyboards and knowledge from film, television, and cartoons, I should be able to devise a first pass at a method for users to control the camera and use this as a base on which to build.

Summary of Expected Research Plan

Subthesis 1: The system should clearly communicate to girls that it can be used for the purpose of telling stories.

- Improve error handling in Stencils.
- Leverage Stencils to create a help system based on observed needs middle school girls

<p>have while writing stories in Alice.</p>
<p><i>Subthesis 2: The system should provide support to help girls generate story ideas and steer them towards kinds of stories with which they can be successful early on.</i></p> <ul style="list-style-type: none"> • Refine design guidelines for Story Kits • Identify storytelling tasks that the Alice system better support through user testing. • Add additional storytelling supports to Alice based on user testing.
<p><i>Subthesis 3: The system should provide commands at a sufficiently high level to allow girls to focus primarily on creating stories rather than requiring them to focus on numerous subgoals.</i></p> <ul style="list-style-type: none"> • Use iterative design and testing to develop and refine animations that allow users to get started writing stories in Alice more quickly across three areas: communicating backstory, blocking characters, and controlling the camera: <ol style="list-style-type: none"> 1. Communicating backstory <ul style="list-style-type: none"> ▪ First step: Add <i>Alice.narrate</i> ▪ Later steps: <ul style="list-style-type: none"> • Explore visual ways to communicate backstory based on styles in films, television, and cartoons. • Add support that will allow users to easily incorporate visual backstory into their own creations. • Add support for multiple scenes in Alice. 2. Blocking characters <ul style="list-style-type: none"> ▪ First step: Add <i>Sally move to Jackie howFar = 1 meter whichDirection = front</i> ▪ Later steps: <ul style="list-style-type: none"> • Develop Locomotion and Emotion based animations that can be applied to existing models in the Alice gallery. • Add supports to Alice that will allow users to more easily create animations that require character-character or character-object interactions. 3. Controlling the Camera <ul style="list-style-type: none"> ▪ Determine requirements for camera motion in Alice stories. ▪ Develop camera controls that will allow users to easily add camera motion to their stories

4. EVALUATION

Ideally, we would like to be able to show that girls introduced to programming through creating animated stories in Alice during middle school are more likely than their peers who have not been exposed to Alice to pursue programming or computer science. This exposure might take a

number of forms: taking a programming class, majoring in Computer Science or starting a computer-related job. Unfortunately, to collect these kinds of numbers is unrealistic; I cannot wait between five and twenty years for results. Also, since this is fairly early work, I need to be able to determine quickly whether or not the approach of using storytelling as a motivating end-goal when introducing programming is likely to be successful in bringing more girls into computer science without investing twenty years in initial explorations. To evaluate storytelling as an approach to introducing programming to middle school girls, I will look for behavioral and attitudinal changes between girls introduced to programming using a traditional approach and girls introduced to programming as a means to tell stories.

Psychology studies in a variety of areas have verified the common belief that if you can make a person have a more positive attitude towards a given activity, that person will be more likely to pursue that activity in the future(Levine and Donitsa-Schmidt 1997). Further, a variety of attitude assessment surveys have been developed and validated for computer science. Rather than tracking girls' behavior with respect to computer science for many years, we will instead try to show that a first exposure to programming through storytelling is more successful than a traditional approach to introducing programming in giving girls a positive attitude towards computers and computer science. Specifically, I will attempt to establish four things:

1. Middle school aged girls do some programming in Alice.
2. Middle school aged girls enjoy programming stories in Alice.
3. Middle school aged girls understand that by creating stories in Alice, they are programming.
4. Middle school aged girls introduced to programming through storytelling have more positive attitudes towards computer science and more confidence that they can succeed in Computer Science than those who were introduced to programming in a more conventional way.

I will assume that if an experience with Alice can positively influence girls' attitudes towards computer science, that they will be more likely to pursue programming and computer science later on. However, since girls will be using Alice for a fairly short time, I may not find significant attitudinal changes; it is rare for anyone to significantly change their attitude about something over the course of a few hours. Despite the rarity, I hope that learning to program through storytelling in Alice will be a life-changing event for a small number of girls.

Although the inspiration for this work comes from the observation that women tend to be interested in the applications of computing, the factors that draw people into a given area are many and complex. I will collect detailed information about the users past experiences with computers, favorite subjects in school, encouragement from parents, etc. I will look for other potential factors that may influence girls' decisions to pursue or not pursue programming.

4.1 Comparison System

To evaluate the efficacy of introducing middle school girls to programming through storytelling in a system that supports storytelling, I will look at how much two factors effect girls' performance with and interest in programming: 1) whether or not the system includes supports for storytelling and 2) whether or not the assignment focuses on programming or storytelling.

The Alice system provides support for program construction not found in other systems that are not the result of this work. When we compare the Alice version that will come from this work against another approach to learning to program, we do not want the results to be influenced by the support for program construction that Alice provides. To avoid this, we will compare a version of Alice modified to support storytelling with a version of Alice that does not include storytelling support. In this comparison, there are two factors we need to pay attention to: the impact of the storytelling task and the impact of the system's support for storytelling. There are four possible combinations of task and system:

	Storytelling-Based Assignment		
Storytelling Support		Yes	No
	Yes	✓	✓
	No	?	✓

I will study cases 1, 2 and 4. Through iterative testing and design, I will be able to create a version of Alice that is more suited to storytelling than Alice version 2.0. In creating the Storytelling version of Alice, I am starting with a Storytelling-based assignment and a system that does not specifically support storytelling. I have done several preliminary user observations of girls attempting to create stories in Alice without storytelling support ; their frustrations at having to create stories out of primitive animations like move and turn lead me to believe that this

approach is unlikely to be successful. As I develop and refine the Storytelling version of Alice, I will attempt to document the problems users encounter in trying to create stories in a system that does not support it. If possible, I will use the data I gather while creating the storytelling version of Alice to cover this case.

In my preliminary investigations in how to make Alice better support storytelling, I have made several changes to the Alice system that are now included in our public releases. However, to measure the effect of Storytelling based changes, I will compare a version of Alice with the supports I develop for storytelling with a version of Alice that does not include any storytelling based modifications.

The two versions of Alice	
Alice without Storytelling Support	Alice with Storytelling Support
Stencils tutorial using non-story based examples No “say,” “think”, or “narrate animations” Full Gallery, but no StoryKits	Stencils tutorial using story-based examples ”say,” “think,” and “narrate” animations Full gallery and story kits Additional storytelling supports may include: Camera controls inspired by film and needs of story authors Methods to more easily move characters into positions so they can interact Better support for animating people’s subparts Emotionally based animations Locomotion based animations

In this work, I hope to demonstrate that by focusing on one particular end goal, understanding the needs of users working towards that end goal, and providing adequate support for that end goal within a programming system, it is possible to give girls a positive introduction to programming. The Storytelling version of Alice will incorporate a number of changes based on feedback from user testing which will all be aspects of adequate support for storytelling. By testing all of these small changes together, I will not be able to isolate the relative impacts of each change. However, it is unlikely that any one of the changes that I make to Alice will have a large effect on how girls’ engagement with the system or that any one of these changes, applied to another

programming system, would substantially impact girls' engagement with it. By studying girls' behavior with the system, and their conversations with each other as they create stories in Alice, I hope to be able to demonstrate that it is possible to design a programming system that is more successful than the traditional presentations of programming at getting girls interested in learning to program and demonstrate that teaching programming as a means to an interesting end, such as storytelling, will be more likely to interest girls in learning to program.

4.2 User Study

I will conduct user studies in the form of workshops. A workshop will consist of three main portions.

1. Users will take a pre-workshop survey to collect information on their background with computers and their attitudes towards computers.
2. Users will learn how to create programs in Alice and will be asked to create their own program. The programs they create will be collected and studied to verify that they did some programming.
3. Users will take a post-workshop survey to verify that they understood they were programming, to find out whether they enjoyed their programming experience, and to again measure their attitude towards computer science.

We might eventually hope to create a semester or year long programming course at the middle school level. Realistically, this is unlikely to happen in the near future. But, using shorter experiences will enable us to go through a greater number of iterations, try a wider variety of ideas with the girls, and adapt our approach based on our experiences with girls. Further, if one cannot get girls through their first hour with a positive experience, it is unlikely that one will get them through a whole course with a positive experience. However, even within shorter experiences there is a large range of potential lengths. Since we will necessarily be developing curricular materials and lesson plans to facilitate these workshops, we will choose the length of our workshop such that it could be useful to groups giving workshops to middle school aged girls around the country. We have identified three non-school based groups that present technology programs to middle school girls: Expanding Your Horizons (one day workshop with several 1-1.5 hour class sessions) (EYH), Girl Scout Programs (typically designed to last between 3 and 6 hours)(GS), IBM's Women in Technology Programs (3 hour sessions), and various technology camps for girls around the country (typically 4-5 1-1.5 hour sessions)(WIT). If Alice were to be used in a school, it would likely begin as a short unit, one or two weeks, presented as part of another course, which would allow for about 4 – 8 hours of instruction. To enable schools and

other organizations conducting units or workshops for middle school girls to use the materials I develop, I will focus on a 6 hour study, but provide lesson plans for a shorter 3 hour workshops as well.

4.3 A Meaningful Introduction to Programming

One of the four goals we stated for evaluation was that the students do “some programming.” The ACM is in the process of creating a curriculum for Computer Science education on the K-12 level (Tucker, Deek et al. 2002). Included in the draft are recommendations of what concepts and activities should be introduced at different ages.

Grades 6- 8

The ACM task force recommends that kids at this age become fluent computer users. They should be able to use a variety of tools in pursuit of other educational objectives. The ACM also makes a case for the inclusion of “Algorithmic Thinking” as an important, and underemphasized part of general education. They recommend that students complete problems like “give a complete algorithmic definition for baking cookies, finding your way out of a maze, or making a sand castle.”

Grades 9 – 10

In early high school students should take a year-long course that is designed to impart general knowledge about computer hardware, software, languages, networks, and their impact in the modern world. This course should include a programming component that introduces the following concepts:

- Variables, data types, and the representation of data in computers
- Managing complexity through top-down design
- Procedures and parameters
- Sequences, conditionals and loops (iteration)
- Tools for expressing design – flowcharts, pseudocode, UML

Grades 11 – 12

In later high school, students should take a year-long course that focuses on developing programming skills, particularly algorithm development, problem solving and programming using software engineering principles. The programming components of this course should include:

- Methods (functions) and parameters
- Recursion
- Objects and classes (arrays, vectors, stacks, queues, and their uses in problem solving)
- Graphics Programming
- Event-driven and interactive programming

In our workshops we will introduce middle school students to the following concepts most of which are on the ACM curriculum for either the 9-10 grade or the 11-12 grade computer science classes:

- Definition of Object (11 – 12 ACM)
- Definition of Method (11 -12 ACM)
- Pseudocode (9-10 ACM)
- Top-down design (9-10 ACM)
- Loops (9-10 ACM), Do In Order, and Do Together

Below is a sample schedule designed to take a total of 6 hours, but broken into 4 1.5 hour sessions. The presentation order and pace was designed based on pilot experiences with students in this age group. I will collect and study the worlds created by girls attending the workshops to assess their mastery of these concepts. By asking our test users to create storyboards, I can gain some insight into their goals and intentions. I will also compare the finished worlds with their storyboards and interview girls about why the changed sections in which the storyboard and finished world differ. This may help to pinpoint other areas where Alice does not have sufficient supports for storytelling.

Day	Concepts	A Workshop	B Workshop
Day 1	1. Definition of Object 2. Definition of Method	1. Survey 2. Intro – “learning to program” 3. Paper-based tutorial 4. Playing with the system 5. Intro to storyboarding	1. Survey 2. Intro – “writing animated stories” 3. Stencils-based tutorial 4. Playing with the system 5. Intro to storyboarding
Day 2	1. Global vs object	1. More methods	1. More methods

	methods 2. Design, pseudocode 3. Organizing complexity 4. Basic methods 5. DoT vs DoIO, loops	2. Basic methods 3. Creating your own 4. World vs Obj methods 5. Organizing your work 6. DoT, If, and Loop 7. Clarify goals -> solve some programming task	2. Basic methods 3. Creating your own 4. World vs Obj methods 5. Organizing your work 6. DoT, If, and Loop 7. Clarify goals -> write a story
Day 3		1. Moving the camera 2. Finish story boards 3. Start implementation	1. Moving the camera 2. Finish story boards 3. Start implementation
Day 4		1. Complete implementation 2. Final survey 3. This is real programming 4. Present Projects	1. Complete implementation 2. Final survey 3. This is real programming 4. Present Projects

The reader may notice that while Workshop A tells students that they will be learning to program at the beginning of the workshop, Workshop B does not mention programming until the end. While most middle school girls do not have a clear notion of what programming is, they do understand that programming is associated with programmers and that girls are usually not programmers. If girls do not realize that building stories in Alice is programming, they will not reject it because it is programming. Once they understand that building stories in Alice is programming, they can evaluate their interest in programming in light of their experience with Alice.

4.4 Curricular Materials

To complete these workshops, we will need two categories of curricular materials: materials pertaining to the system and materials pertaining to the approach.

Curricular Materials about the System

As much as possible, the materials pertaining to the system should be identical and should be delivered either by video or paper to ensure consistency between classes. Where it is not possible to share materials, the materials for each condition should use the same concepts and require the same level of effort from the students.

Curricular Materials about the Approach

The materials for the traditional and storytelling-based approaches will necessarily be different in several places in the workshop: the introduction, tutorial, and assignment.

4.4.1 Introduction

At the beginning of the workshop, the instructor will welcome the students and say a few words about the goals for the workshop.

- Traditional: This workshop is about learning to program.
- Storytelling: This workshop is about building animated stories.

4.4.2 Tutorial

Before beginning to create worlds in Alice, the students will need to learn how to use the Alice system. Both tutorials should present the same concepts in the same order.

- Traditional: This tutorial is about learning some of the basic constructs in programming. Examples should demonstrate programming constructs in simple contexts.
- Storytelling: This tutorial is about learning how to use the features that Alice provides to create stories. Examples should show how programming constructs would be used within the context of telling a story.

4.4.3 Task

After completing the tutorial, students will be asked to create their own worlds. The programming task can vary along at least two dimensions: motivation (programming or storytelling) and how much freedom students are given in what they can create (one correct answer to infinite numbers of answers). In this study, we would like to isolate the contribution of the motivation behind the programming assignment as much as possible. Perhaps the most obviously fair way is to make both assignments have one and only one solution. While this is not uncommon in teaching programming, part of the advantage of a storytelling-based approach is that children can work on stories they find motivating and feel a real sense of ownership over their project. A single pre-created story is unlikely to be relevant to all students and they are unlikely to feel that the work is their own. The other end of the spectrum is more open ended: an infinite number of worlds are possible that meet specifications like “demonstrate how to use the following 3 constructs” or “tell

a story but be sure to include these elements.” It can be hard to quantify exactly how much freedom you give students with assignments like these. The set of all story-based worlds is a subset of the set of all worlds. Would that make one assignment in the space of all worlds more free than one in the space of all stories? Perhaps. To create these assignments, we will choose two infinite subsets of the space of all worlds and impose the same constraints on them.

- Traditional: Create an Alice world that demonstrates the use of the following programming constructs: character method, loop, DoInOrder, and DoTogether.
- Storytelling: Create an animated story in Alice. In creating your story you should use each of the following programming constructs at least once: character method, loop, DoInOrder, DoTogether

4.4.4 This is Real Programming

After completing work on their Alice worlds, an instructor will tell both groups that what their experience building worlds in Alice is similar to programming in languages like Java and C++ in that programmers instruct objects to perform various actions, use the same control structures that they used in Alice, and create methods to organize their programs. The instructor will tell them that when using languages like Java and C++, programmers have to type their programs rather than dragging and dropping tiles and that not all programs use 3D graphics.

4.5 Data Collection

Earlier I stated five objectives:

1. Show the girls did some programming.
2. Show that girls introduced to programming through storytelling enjoyed their programming experience.
3. Show that the girls understood that they were programming.
4. Show that girls introduced to programming through storytelling have more positive attitudes towards computer science than those who were introduced to programming in a more conventional way.

I will assess my success in achieving these four objectives using two primary instruments: surveys and the worlds the girls create during the workshops.

4.5.1.1 ESTABLISH THAT THE GIRLS DID SOME PROGRAMMING

I will collect and analyze the worlds created by girls during the workshops. Specifically, I will look for the usage of character methods, DoInOrders, DoTogethers, Loops, and If statements as specified in the assignments given to the students. Further, I will look for differences in the worlds created by students in traditional workshops and those created by students in storytelling based workshops such as the numbers of different programming constructs used, numbers of lines of code, usage of programming and constructs not covered in the workshop.

4.5.1.2 ESTABLISH THAT GIRLS ENJOYED THEIR PROGRAMMING EXPERIENCE

While it is hard to quantify enjoyment, I intend to include a few questions in the post-workshop survey that are intended to elicit whether or not the students enjoyed their experiences. These questions will take two forms: direct questions about their feelings about or intended usage of Alice and indirect questions that we expect to elicit a few types of answer. A direct question might be to ask whether students plan to continue using Alice at home. An indirect question might be to ask students to write a few sentences to convince a friend to learn to use Alice. The answers to this question will probably take one of two forms: 1) it's fun or 2) it's good for you.

In addition to querying the students, I will also look for behavioral differences between groups of students using the storytelling and non-storytelling versions of Alice such as how much time they spend creating their worlds, how many lines of code they write, and how long they continue to work on their worlds after they are no longer required to be working. Alice provides some basic support for analyzing the content of worlds, which is extensible to meet my needs. Since many of the Alice workshops will be given through organizations (e.g. homeschool groups, girl scout troops, etc) I may get requests for subsequent workshops if enough students in the group enjoyed their experience with Alice.

Another indicator of girls' interests in Alice is whether they install it on their home machines and continue to use it. At the end of each workshop I will give students information about an Alice website where they can download Alice, post worlds that they create, and view worlds that others have created. Each workshop will have a unique web page and I will instrument these websites to track the numbers of visits, downloads etc.

4.5.1.3 ESTABLISH THAT GIRLS UNDERSTOOD THEY WERE PROGRAMMING

The post-workshop surveys will also include a couple of questions designed to assess whether students understood that they were programming. Example questions might be:

- Describe what you did today
- How do you write a program in Alice?

4.5.1.4 ESTABLISH THAT GIRLS HAVE A MORE POSITIVE ATTITUDE TOWARDS COMPUTER SCIENCE

There are a variety of attitude surveys that have been developed and validated. The Computer Attitude Survey (CAS) is one of the most commonly used instrument in measuring attitudes towards computers (Nash and Moroz 1997). It has been used both to measure general attitudes towards computers, and to assess the efficacy of various types of training in changing peoples attitudes towards computers (Gunter and Gunter 1984; Massoud 1991; Mertens and Rabiou 1991; Pope-Davis and Twing 1991; Bennett 1995). In its original form, the CAS measured attitudes along four subscales: Computer Confidence, Computer Liking, Computer Anxiety, and Computer Usefulness. Recent Research suggests that the computer confidence and computer anxiety subscales are measuring the same thing (Nash and Moroz 1997). I initially plan to use the version of the CAS proposed by Nash and Moroz, which combines the confidence and anxiety subscales and adds another subscale that corresponds to attitudes towards academic endeavors associated with computer training. Students will take the Nash and Moroz CAS before and after the workshop. To factor out the impact of taking the same survey twice, I will ask a control group of girls who will not participate in any Alice workshops to take the survey two times, separated by the length of the workshop.

I will add questions to the post-survey to specifically examine students' interest and confidence in continuing to use Alice and other programming related activities.

4.5.1.5 ADDITIONAL INFORMATION

The pre-workshop survey will ask students to provide some information about their prior programming and computer experiences.

Preliminary pre and post surveys can be found in Appendix A.

5. RELATED WORK

5.1 *Programming Environments and Languages for Novice Programmers*

Over the past 40 years, many researchers have attempted to make programming accessible to a broader range of people by reducing the difficulty of getting started learning to program. I have completed a comprehensive survey of programming systems and environments designed for novice programmers (Kelleher and Pausch 2003). While it is not reasonable to discuss all of the systems here, I will briefly summarize the main approaches that researchers have used in attempting to make programming accessible to a broader range of people.

Learning to program consists of two main tasks: learning to form a syntactically correct program and learning to use logic and control structures to solve problems. A novice learning a general-purpose language like C++ or Java must tackle these two challenges simultaneously; most programming systems for novices attempt to simplify or provide additional support for one of these two tasks. The general philosophy behind these systems is that if the process of learning to program is less painful for novices, more people will be interested in learning to program.

5.1.1 *Simplify Textual Programming Languages:*

Textual programming languages for novice programmers often remove any unnecessary syntax, limit the size of the language, and choose understandable names for commands. A typical language in this category has a small number of carefully chosen constructs and commands and fewer syntactic requirements (e.g. parenthesis, semicolons, commas, etc) than most general-purpose languages.

5.1.2 *Provide Alternatives to Typing Programs:*

Despite the attempts to make programming languages simpler and more understandable, many novices still struggle with syntax: remembering the names of commands, the order of parameters, whether or not they are supposed to use parentheses or braces, etc. Rather than starting with a textual programming language and attempting to improve it, these systems attempt to provide novices with non-textually based methods for creating programs. There are two common techniques. The first technique allows novices to connect graphical or physical objects representing elements of a program such as commands control structures or variables. By moving and combining graphical or physical objects, novices can create programs. The second technique

allows novices to create programs by selecting from a concrete set of actions in an interface, often by pushing a sequence of buttons or filling in a form.

5.1.3 Helping Students Visualize Programs:

Most program execution is invisible: values in variables change and perhaps the program occasionally prints to the screen. But, if the program does not behave as the programmer anticipated, it can be difficult to determine what is going wrong. The systems in this category try to help students understand what happens during the execution of programs, either by placing programming into a concrete setting so that students can more readily imagine what the execution of a program “looks like” or by providing a simulation to allow students to watch their programs execute.

5.2 Addressing Sociological Barriers

While it is helpful to make the beginning steps of learning to program as painless as possible, it has not been sufficient to drastically change the number or kinds of people that enroll in programming classes or choose to pursue computer-related careers. The difficulty of learning to program is not the only reason that people choose not to pursue programming, either in school or as a career. Some people do not immediately see a compelling reason to learn to program. Some believe that programmers work almost exclusively with computers and almost never with other people. Some feel that there would be negative social consequences to learning to program. To truly broaden the ranks of Computer Science, we need to understand and address some of these sociological issues in people’s first exposures to Computer Science. A few programming environments have begun to address sociological concerns.

5.2.1 Social Context:

Several systems provide programming environments that allow users to either work together on a shared program or to easily share and build upon one another’s work. While not directly addressing the concerns relating to computing careers being isolated, they demonstrate to novices that programming can be a collaborative effort and that creating interesting programs can result in positive feedback from peers.

5.2.2 End-Goals for Programming:

For students who are not inherently interested in the computer as a machine, it is crucial to answer the question “Why should I program?” A few children’s games and children’s programming systems have attempted to embed programming within a “fun” context. For many

of the games, programming is presented either as a way to solve a particular puzzle and move on to the next game level or as a way to participate in cyber-battles.

5.3 Storytelling Software

Creating software that supports children in telling stories is not a new idea, there have been a number of commercial storytelling programs developed for children, some specifically targeting girls. Recent storytelling programs for children include Barbie Storymaker, Storymaker, and the American Girls Premiere (LC 1999; Mattel 2002; 2003). These kinds of programs typically provide a limited selection of animations and do not teach programming.

Play, a programming system for novices was designed to support both storytelling and programming (Tanimoto and Runyan 1986). However, this program was designed for pre-literate children, supports a limited number of programming constructs, and only provides support for animations through quickly switching between images.

(2003). StoryMaker, SPA Software.

AAUW (1992). How Schools Shortchange Girls: A Study of Major Findings on Girls and Education. New York, NY, Marlowe & Company.

AAUW (1996). Girls in the Middle: Working to Succeed in School. Washington, DC, American Association of University Women Educational Foundation.

AAUW (1998). Gender Gaps: Where Schools Still Fail Our Children. Washington, DC, American Association of University Women Educational Foundation.

AAUW (2000). Tech-Savvy: Educating Girls in the New Computer Age. Washington, DC, American Association of University Women Educational Foundation.

Begel, A. (1996). LogoBlocks: A Graphical Programming Language for Interacting with the World. Electrical Engineering and Computer Science Department. Boston, MA, MIT.

Bennett, C. K. (1995). "A Staff Development Partnership for Technology Integration." Journal of Staff Development **16**(3): 19-22.

Brunner, C., D. Bennett, et al. (1998). Girl Games and Technological Desire. From Barbie to Mortal Kombat. J. Cassell and H. Jenkins. Cambridge, MA, MIT Press: 72-88.

Castell, S. d. and M. Bryson (1998). Retooling Play: Dystopia, Dysphoria, and Difference. From Barbie to Mortal Kombat: Gender and Computer Games. J. Cassell and H. Jenkins.

CAWMSET (2000). Land of Plenty: Diversity as America's Competitive Edge in Science, Engineering and Technology. Arlington, VA, Commission on the Advancement of Women and Minorities in Science, Engineering, and Technology.

Collins, W. A. and S. A. Kuczaj (1991). Developmental Psychology: Childhood and Adolescence. New York, NY, Macmillan.

- Conway, M. (1997). Alice: Easy-to-Learn 3D Scripting for Novices. School of Engineering and Applied Science. Charlottesville, VA, University of Virginia.
- Cooper, S., W. Dann, et al. (2003). Teaching Objects-first In Introductory Computer Science. SIGCSE, Reno, NV.
- Drucker, S. M. and D. Zeltzer (1995). CamDroid: A System for Implementing Intelligent Camera Control. ACM Special Interest Group on Computer Graphics and Interactive Techniques.
- EYH Expanding Your Horizons.
- Finzer, W. and L. Gould (1984). Programming by Rehearsal. Palo Alto, CA, Xerox Palo Alto Research Center.
- Frankel, L. (2002). Research Facts and Findings: Identity Formation in Adolescence.
- Gindling, J., A. Ioannidou, et al. (1995). LEGOsheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick. Visual Languages, Darmstadt, Germany.
- Gleicher, M. and A. Witkin (1992). Through-the-Lens Camera Control. ACM Special Interest Group on Computer Graphics and Interactive Techniques.
- Gray, N. Why have "high-level" languages?
- Grunwald (2003). Children, Families and the Internet 2003, Grunwald Associates.
- GS Computer Fun (Girlscout Badge).
- Gunter, R. E. and G. A. Gunter (1984). The Effect of Class Size on Student's Attitudes toward Computers, ERIC Document Reproduction Service.
- Guzdial, M. (2003). A Media Computation Course for Non-Majors. ITiCSE, Thessalonika, Greece.
- Hartmann, W., J. Nievergelt, et al. (2001). Kara: finite state machines, and the case for programming as part of general education. Symposia on Human Centric Computing 2001, Stresa, Italy.
- He, L.-w., M. F. Cohen, et al. (1996). The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. ACM Special Interest Group on Computer Graphics and Interactive Techniques.
- Hecker, D. E. (2001). Occupational employment projections to 2010, Office of Occupational Statistics and Employment Projections, U.S. Department of Labor Statistics.
- Honey, M., B. Moeller, et al. (1991). Girls and Design: Exploring the Question of Technological Imagination. New York, Center for Children and Technology.
- ITAA (2001). When Can You Start?: Building Better Information Technology Skills and Careers, Information Technology Association of America.
- Kahn, K. (1996). "Drawings on napkins, video-game animation, and other ways to program computers." Communications of the ACM **43**(3): 104-106.
- Kay, A. Etoys and Simstories in Squeak.
- Kelleher, C. (2002). Alice Stencils Tutorial.
- Kelleher, C. and R. Pausch (2003). Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers. Pittsburgh, PA, Carnegie Mellon University.
- Knabe, K. (1995). Apple Guide: a case study in user-aided design of online help. Conference on Human Factors in Computing Systems, Denver, Colorado.
- LC (1999). The American Girls Premiere, The Learning Company.

- Levine, T. and S. Donitsa-Schmidt (1997). "Commitment to Learning: Effects of Computer Experience, Confidence, and Attitudes." Journal of Educational Computing Research **16**(1): 83-105.
- Lionet, F. and Y. Lamoureux (1994). Klik and Play, Maxis.
- Logo Computer Systems, I. (1995). My Make Believe Castle.
- Logotron (2002). Magic Forest.
- Marcia, J. (1980). Identity in Adolescence. Handbook of Adolescent Psychology. J. Adelson. New York, NY, Wiley - Interscience.
- Margolis, J. and A. Fisher (2002). Unlocking the Clubhouse: Women in Computing. Cambridge, MA, MIT Press.
- Massoud, S. L. (1991). "Computer Attitudes and Computer Knowledge of Adult Students." Journal of Educational Computing Research **7**(3): 269-291.
- Mattel (2002). Barbie StoryMaker, Mattel Media.
- Mertens, D. M. and J. Rabiou (1991). The Influence of Computer Experience on Attitudes and Learning for Preservice Deaf Teachers. Annual Meeting of the American Educational Research Association, Chicago, Illinois.
- Miller, P., J. Pane, et al. (1994). "Evolution of Novice Programming Environments: The Structure Editors of Carnegie Mellon University." Interactive Learning Environments **4**(2): 140-158.
- Muir, D. (2001). Adapting Online Education to Different Learning Styles. National Educational Computing Conference.
- Nash, J. and P. Moroz (1997). "An Examination of the Factor Structures of the Computer Attitude Scale." Journal of Educational Computing Research **17**(4): 341-356.
- NCES (1998). State requirements for high school graduation, in Carnegie units: 1993 and 1996, National Center for Education Statistics. **2003**.
- NCES (2002). State requirements for high school graduation, in Carnegie units: 2001, National Center for Education Statistics. **2003**.
- Newburger, E. C. (2000). Home Computers and Internet Use in the United States: August 2000, U.S. Census Bureau.
- Oviatt, S. (2000). "Taming Recognition Errors with a Multimodal Interface." Communications of the ACM **43**(9): 45-51.
- Pope-Davis, D. B. and J. S. Twing (1991). "The Effects of Age, Gender and Experience on Measures of Attitude Regarding Computers." Computers in Human Behavior **7**: 333-339.
- Repenning, A. (1993). Agentsheets: a tool for building domain-oriented visual programming. Conference on Human Factors in Computing Systems.
- Roper (1999). The America Online/Roper Starch Youth Cyberstudy.
- Smith, D., A. Cypher, et al. (1997). KidSim Programming Agents without a Programming Language. Proceedings of 6th European Conference on Software Engineering.
- Stebbins, G. (1902). Delsarte System of Expression. New York, Edgar S. Werner Publishing and Supply Co.
- Stone, L. J. and J. Church (1984). Childhood and Adolescence: A Psychology of the Growing Person, 5th Edition. New York, Random House.

- Tanimoto, S. and M. Runyan (1986). Play: An Iconic Programming System for Children. Visual Languages. S. K. Chang, T. Ichikawa and P. A. Ligomenides, Plenum Publishing Corporation: 191-205.
- Tomlinson, B., B. Blumberg, et al. (2000). Expressive Autonomous Cinematography for Interactive Virtual Environments. Fourth International Conference on Autonomous Agents, Barcelona, Spain.
- Tucker, A., F. Deek, et al. (2002). A Model Curriculum for K-12 Computer Science: Report of the ACM K-12 Education Task Force Computer Science Curriculum Committee, ACM.
- WIT IBM Women in Technology Program.
- www.alice.org (2003). Alice, Carnegie Mellon University.