This material was developed as part of three projects funded by the National Science Foundation – through grants ESI-0624549, DRL-0755381 and DRL-0909733, which examine whether Storytelling Alice could be used by middle school students to create games in both pairs and as individuals and whether it showed some potential for building IT fluency and computational thinking.
# Table of Contents

Introduction........................................................................................................................................... 1

*Storytelling Alice* and its Predecessor, *Alice* ................................................................. 1

Using *Storytelling Alice* to Teach Game Development................................................................. 2

The Pros and Cons of Using *Storytelling Alice* for Game Development ..................................... 3

About the *Storytelling Alice* Game Development Program.......................................................... 4
  
  Suggested Sequence of Activities ......................................................................................... 5
  
  Altering the Program ............................................................................................................ 6

Getting Started ........................................................................................................................... 7

Important Links, Additional Resources and Contacts for More Information.............................. 9
**Storytelling Alice Game Development Program**

**Introduction**

The material included here consists of an outline for a 30-hour class and supporting materials for teaching late elementary, middle-school and early high school students how to make games using *Storytelling Alice* (SA), a 3D virtual-world programming environment. The lessons are based on a set of challenges, each designed to teach a particular programming skill or set of skills. Formatted as checklists, these provide the students with increasing autonomy as they hone their skills and gain confidence.

This material was developed as part of three projects funded by the National Science Foundation – through grants ESI-0624549, DRL-0755381 and DRL-0909733. In these projects, we explored (and are exploring) whether SA could be used by middle school students to create games in both pairs and as individuals and whether it showed some potential for building IT fluency and computational thinking. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

**Storytelling Alice and its Predecessor, Alice**

*Storytelling Alice* (SA) is an open source 3D virtual-world programming environment that was adapted from *Alice* in 2006/2007 by Caitlin Kelleher as part of her doctoral work in Computer Science. The original *Alice* was developed at Carnegie Mellon University and designed to be a student’s first exposure to object-oriented programming. It allows students to learn fundamental programming concepts and to create animated stories and games by dragging and dropping graphic tiles. The result is a set of programming instructions that correspond to standard statements found in a production-oriented language such as Java. This method helps to prevent frustrating syntax errors because students can only select from a list of valid parameters and operations. Also, feedback is immediate; students can instantly see how their animation programs run and easily understand the relationship between the programming statements and the behavior of objects in their animation by simply clicking the play button.

The original *Alice* and SA are quite similar; in fact, they look identical at first glance. However, there are several important differences. *Alice* is used primarily by high school and college students whereas SA was designed specifically for middle school students. It is particularly appealing to girls in this age group because it includes additional characters that appeal to middle school youth (e.g., a lunch lady) and a variety of pre-programmed animations and behaviors that enable users to easily program social interactions and help to generate story ideas. Also, SA includes some functionality that is not available in *Alice*. Therefore, SA worlds currently do not run in *Alice* although *Alice* worlds will open and play and can even be modified in SA. Finally, unlike *Alice*, which is multi-platform, SA is PC-based only; it will not run on Mac or Linux-based computers. New versions of *Alice* are in the making that will have the features of SA that are specifically useful for middle school students.

Using *Storytelling Alice* to Teach Game Development

SA has proven to be an effective way to motivate preteens to learn basic programming. Kelleher’s study comparing middle-school girls’ experiences with learning to program in *Storytelling Alice* and the original version of *Alice* showed that users of SA spent 42% more time programming, and despite the focus on making programming more fun, they were just as successful at learning basic programming concepts. ([http://www.alice.org/kelleher/storytelling/](http://www.alice.org/kelleher/storytelling/))

However, as the name implies, Kelleher’s design and research efforts focused on SA’s storytelling features. To determine if SA could be used by middle school students to make games rather than stories, ETR Associates conducted a National Science Foundation-funded feasibility study in the Summer of 2008. This study consisted of two classes that taught middle school students, both boys and girls, how to make games using SA. These classes ran simultaneously at community centers, one in a small, predominantly White city and the other in a nearby agriculturally-based, predominately Latino, community. The materials were tested and refined as part of the Girl Game Company program, created by ETR Associates with funding from the NSF. To see some of the games created by the youth in these classes, go to: [http://psweb.etr.org/alice/](http://psweb.etr.org/alice/)

The results of this pilot study suggest that SA can be used by middle school students to make games. Twenty-eight students out of 43 original participants completed the course, and 23 SA worlds with two or more game-defining qualities (e.g., the existence of rules, an uncertain outcome, goals, player interactions that affect outcome, or player investment in the outcome) were made in a two-week, two-hour-per-day period. Fourteen of these were made by pairs. The games ranged from those that tested knowledge to races against a clock and showed that students did engage in key aspects of IT fluency. All the games included player interaction (called “events”), and nearly half (48%) included programmer-defined methods (i.e., programming that tells an object how to behave). More than half (52%) included parallelism (i.e., two things are programmed to happen simultaneously), and 39% used programmer-defined parameters and variables. The students were very proud of their creations and anxious to have others play them.¹

Based on the results of this pilot study, the National Science Foundation awarded ETR Associates a three-year grant in early 2009 to study computational thinking and computer game design during middle school. This most recent study, which is known as the iGame Project, is addressing three specific research questions:

1. What is a developmentally appropriate definition of computational thinking in middle school?
2. Does creating a computer game promote computational thinking?
3. Under what conditions does pair programming produce greater gains in computational thinking than solo programming?

In the iGame Project as in the pilot study, middle school students in informal afterschool settings are using *Storytelling Alice* to make games, and they are learning this programming environment through a self-paced, challenge-based system that has been refined according to lessons learned from the previous projects. All students are voluntary participants and include both boys and girls. To study solo versus pair programming, classes offered at each of three schools use either pair or solo programming in the Fall semester and the other type of programming in the Spring.

**The Pros and Cons of Using *Storytelling Alice* for Game Development**

**Pros**

1. **Error-free code is easy to produce and understand.** SA’s interactive interface allows users to drag and drop graphic tiles to create a program. The result is a set of syntactically correct instructions in a language that reads like standard sentences. Because this interface makes it almost impossible for the programmer to make a syntax error, he/she can focus on the logic of the program rather than on small but usually crucial details like a missing semi-colon.

2. **Students get instant feedback.** With the click of the “play” button, students can easily see the results of their efforts and then make immediate changes.

3. **3D and pre-programmed animations make it exciting and fun.** Although SA games are not as past-paced and sophisticated as commercial computer games, the 3D nature of SA and the appealing characters with a variety of pre-programmed behaviors give them a higher level of sophistication than those created in two-dimensional applications.

4. **Code is easy to use for trouble-shooting and problem-solving.** Students can easily export and print the code of sample challenges and games and compare it to code that is not working correctly.

5. **SA is good for certain types of popular games.** Given its 3D nature and the gallery of interesting animated characters, this software is particularly suited for games involving movement, spatial manipulation and social interaction, such as adventure games and scavenger hunts. Since the graphic interface allows programmers to easily generate if/else statements, loops and other statements involving user interaction, it is also conducive to creating games that test players’ knowledge.

6. **With SA, students spend more time programming their games instead of formatting.** Because SA comes with ready-made backgrounds and animations, students can concentrate on game programming rather than on creating or importing backgrounds and characters or changing fonts and colors.

7. **The objects’ built-in animations can stimulate story and game ideas.** Fairies can flap their wings as they fly; ninjas can kick; a lion roars and jumps – all of which can spark the creative process.
Cons

1. **3D can be confusing.** SA’s camera controls allow the user to turn objects and change perspective 360°. Students can also move objects off the screen or make them so small or so far away that they are invisible. It’s therefore easy to lose objects or perspectives.

2. **SA doesn’t handle collision control very well.** Even though there is built-in programming to detect and control object proximity, the 3D element makes it tricky. SA is therefore not the best choice for shooting, drag & drop and maze games.

3. **The gallery has limited objects.** SA has far fewer objects/characters and backgrounds than Alice in its gallery. Objects and backgrounds can easily be imported from Alice, but most do not include built-in animations.

4. **Usability is limited to PCs.** Unlike Alice, SA does not work on any other platform than a PC.

5. **Objects cannot be easily created or customized.** Unlike some other programs, characters and other objects cannot be drawn or imported as clip art and radically modified in SA. And while it is not easy to build 3D models in either SA or general Alice, users can create custom male and female characters using “hebuilder” and “shebuilder” in Alice.

About the Storytelling Alice Game Development Program

This 30-hour program is based on material developed for the pilot feasibility study. They were later refined for the Girl Game Company and the iGame projects. It is suitable for late elementary, middle-school and early high school students primarily in an extended learning, after-school or summer school environment, but it could also be used in the classroom. It is meant to be used by teachers and other education professionals who are generally “tech savvy. Prior experience in teaching programming and game design and using Storytelling Alice would be helpful but is not absolutely necessary.

This material is designed to be as practical as possible and to be used “out of the box” with little or no modification. It begins with 8 to 10 hours of challenges that teach basic SA programming concepts and skills. These include: a) learning about the main areas and windows such as the scenes window, the details panel and the gallery; b) practicing how to add characters and manipulate their actions; c) using the camera controls; and d) creating new methods and changing characters’ properties.

The second half of the course concentrates on game development. It begins with a unit on the difference between stories and games and an introduction to storyboarding. In subsequent units, students learn about and practice how to use the event handlers that control user interaction as they develop their own games.

Several of the graduated challenges on which the program is based are modifications of those in a textbook by Shelley Cashman Herbert. We modified their challenges to make them suitable for a

---

middle school audience using SA rather than Alice and formatted them as checklists (a useful innovation that was suggested by a middle school girl). We also adapted Alice tutorials created by another project funded by the National Science Foundation and developed additional challenges on specific game elements such as timers and counters.

For the iGame project, we added a series of bonus challenges that provide additional practice on key skills already covered or introduced skills that are not absolutely necessary for making a game (e.g., sound). We encouraged the students who moved quickly through the challenges to do all of the challenges in numerical order – both the main ones and the bonuses – and asked those who moved at a slower pace to skip the bonuses and return to them later if time permitted. That strategy helped to maintain class continuity and minimize wide discrepancies in student progress.

The challenges proved to be an effective way of teaching programming and game development to middle school students. As documented through student and teacher interviews and observation, the challenges were highly motivating, allowed students to move at their own pace, and promoted both creativity and independent problem-solving. As one student said, “(What I like best about SA) are the challenges. We had to figure it out. I like doing that.”

In addition to sample games, most of which were created by students, a completed version of each challenge, along with its code, is included in the information presented here. Students can not only use these as models but also compare their code with the samples when they run into problems. Several of the challenges also include a starter world so students do not have to take the time to program elements they have either already learned and practiced or are not yet ready to tackle.

Finally, for those who want more, we have included links to Alice-based tutorials and lesson plans, which can be adapted to SA, as well as other relevant information, such as papers on pair programming.

**Suggested Sequence of Activities**

**Weeks 1 - 4:**
- Introduction to Storytelling Alice, methods
  - Students play sample games and worlds
  - Students take tutorials
  - Teacher provides overview of what students will learn in required and bonus challenges 1-4
  - Begin challenges

**Weeks 5 – 8:**
- More on methods, introduction to events
  - Teacher provides overview of what students will learn in required and bonus challenges 5-8
  - Students continue on challenges
  - Students share their work

---

Weeks 9 – 12:
More on events, storyboarding, game elements, variables and other advanced elements, introduction to student games
• Teacher provides overview of what students will learn in required and bonus challenges 9-13
• Students continue on challenges
• Teacher and students review/play various types of games
• Students begin planning their games
• Students share their work

Weeks 13 – 15:
Variables, if/else, user input, work on own games
• Teacher provides overview of what students will learn in remaining challenges, but makes their completion optional
• Students work on their own games and complete any challenges that teach them elements they want to include in their games.
• Students share their final games

Altering the Program
If you need to alter the 30-hour, two-hour per session plan, consider the following:

• If you have to fit the lessons into 1-hour or 45-minute time slots, you should add at least 10 minutes at the beginning and end of each session for review of the previous session and wrap up of the current session. This is particularly important if the class does not meet every day.

• If you do not have 30 hours, you may eliminate the bonus challenges altogether or give them to students who move quickly ahead of their peers. You might also consider eliminating the following main challenges:
  o Challenge 4, the Chicken Farmers Association Ad Challenge, which teaches about the camera controls and provides additional practice.
  o Challenge 6, the Penguin Dance Challenge, which provides additional practice using methods and covers other elements used later.
  o Challenges 11 Interactive Amusement Park Challenge, which provides additional practice using events.
  o Challenge 15, the Princess Adventure Game Challenge, which provides practice in all the skills previously learned.

• If you have very limited time, you might consider covering one type of simple game only. You could, for example, have the students make a matching game (Challenge 14), or some type of game that uses similar skills, and then eliminate all the subsequent challenges. Although you can still pare down the challenges as noted above, most students will need 8 to 10 hours to learn
the basics by going through the tutorials and the key challenges that lead up to the matching game challenge.

Getting Started

A. Learn about *Storytelling Alice* (SA) and the program

1. **Download Storytelling Alice.** Go to [http://www.alice.org/kelleher/storytelling/download.html](http://www.alice.org/kelleher/storytelling/download.html) and download a free copy of SA to your computer. Remember, unlike *Alice*, SA only works on a PC.

2. **Download the program materials.** Download the following:
   - Sample games and challenges (SA worlds)
   - Handouts – challenges, code (pdf documents)

3. **Take SA tutorials.** Familiarize yourself with SA by going through the three tutorials that come with it. To access them, open SA and then click the “tutorial” button in the “Welcome to Alice” window that comes up. Each tutorial should take no more than 10 to 15 minutes to complete.

4. **Take an additional tutorial from Alice.** Go through a fourth tutorial to learn about user-initiated events. To access this tutorial, take the following steps:
   - After decompressing, open the *Alice* folder and go to: Required>tutorial. Copy “Tutorial3.stl” to your desktop and rename it “Tutorial4.stl” since there is already a Tutorial3 in *Storytelling Alice*.
   - Open the tutorialWorlds folder in the Required folder, select the “penguinChorus1b.a2w” file and copy it to your desktop.
   - Close the *Alice* folder and open your *Storytelling Alice* folder.
   - Drag the “Tutorial4.stl” file on your desktop into SA’s Required>tutorial folder and the “penguinChorus12b.a2w” file into SA’s Required>tutorialWorld folder.
   - When you open SA, the penguin tutorial should be there.

5. **Play sample games.** To familiarize yourself with the types of games that have been created in SA, download, open and play the games that have been created by students and staff. The games created by middle school students can be found at: [http://psweb.etr.org/alice/](http://psweb.etr.org/alice/). To access these SA worlds once you have downloaded them, open SA and then go to “File”> “Open World.”
6. **Do the challenges.** Familiarize yourself with the program and learn more about SA by completing all of the challenges. So that you will be prepared when students have questions, note any problems or confusion you encounter as you work. Save the resulting SA worlds you create. If you wish, you can include your finished products in the samples to show the students. The shorter challenges, especially the earlier ones, should take 10 to 15 minutes to complete. The longer, more complicated challenges may take 30 to 45 minutes.

7. **Create your own game.** If this is your first time using SA, it is important to make at least one of your own games in order to apply what you learned from doing the challenges.

8. **Plan out your class sessions.** If you need to modify your class plan to fit a schedule other than 30 hours and two hours per session, plan your class sessions according to the suggestions for altering the program above.

B. Set up the learning environment.

1. **Make sure there are enough working computers with the proper software and working space.** Given the hands-on nature of this course, it’s important that students have access to computers loaded with working versions of SA (or access to class folders on a network). It is also important that all students have room to work and lay out written materials as well as dedicated spaces to store their work. You do not need a separate computer for each student. Having students work in pairs not only saves computer and work space but enhances the programming process. For more information about pair programming, see “Using Pair Programming” in the Teachers’ Lounge filing cabinet.

   After the computers are set up, open SA on each one and make sure it is working properly. At this time, you should also add the fourth tutorial to each SA folder as noted above under “Getting Started” Step A4.

2. **Set up and communicate user protocols.** Working with a computer technician if necessary, write up protocols for how to log in and out, where and how to access and save files, etc. Make this information easily accessible to students via a handout or poster of the steps.

3. **Prepare materials.** Make copies of the challenges and other handouts for all students. If possible, duplicate the challenges in color. Color graphics are not only more appealing, but the challenges are easier for students to follow when the pictures look like what they see on the screen. Make multiple copies of the code for each challenge and sample game, so that more than one student or pair can use them for problem-solving at one time. Laminate if possible, so students can check off the steps and then erase them if they need to repeat the challenge.
Important Links, Additional Resources, and Contacts for More Information

Important Links

- To download *Storytelling Alice*, go to: [http://www.alice.org/kelleher/storytelling/download.html](http://www.alice.org/kelleher/storytelling/download.html)

Additional Materials

  Information from a National Science Foundation-funded project for integrating *Alice* (not *Storytelling Alice*) into middle schools and high schools in Durham, NC. Includes lesson plans, tutorials and more.
  Links to *Alice* instructional materials, tutorials, publications, forums and more.
- For more information about pair programming, see “Using Pair Programming” in the Teachers’ Lounge resource cabinet.

Contacts for More Information

If you have any questions about the research projects on which these materials are based, contact Jill Denner or Linda Werner:

Jill Denner  
ETR Associates  
831-438-4060 ext. 264  
jilid@etr.org  
Linda Werner  
University of CA, Santa Cruz  
linda@soe.ucsc.edu

If you have any questions about this material, contact:

Pat Rex  
ETR Associates  
831-438-4060 ext. 106  
patr@etr.org