Use of ALICE programming language to aid reading comprehension and to introduce computer science skills in elementary school students

Mercedes R. Lopez
CRA-DMP Intern, Summer 2007
Department of Computer Science, Duke University

Introduction

The demand for computer scientists is growing much more rapidly than schools are producing qualified candidates to fill the growing industrial need. Interest in computer science as a major is declining among incoming students nationwide; the percentage of college freshmen that listed computer science as their most likely major dropped 70 percent from 2000 to 2004.[1] The statistics regarding females are even more staggering. Although there are more females than males enrolled in colleges, only 28 percent of bachelor degrees in computer science are awarded to females.[1] Many studies have indicated that most students who drop the major, do so during the freshman year, and that the number of students remaining in the major decreases drastically with every successive year in college.[2] The demand for computer scientists will continue to be a problem until the causes for the decline in student interest have been addressed. Causes for the decline in the number of students graduating with computer science degrees have been attributed to a number of factors, such as students entering the major without previous programming experience.[3,4]

Illiteracy in America is growing at an alarming rate. As of 2003, the figures, reported by the U.S. Department of Education as a result of the National Adult Literacy Survey,[5] include:

- 42 million American adults cannot read;
- 50 million American adults read at only fourth or fifth grade levels;
- 25 percent of American high school graduates read at less than an eighth grade level;
- 20 percent of all graduating high school seniors are functionally illiterate;
- the number of functionally illiterate adults increases by approximately 2.25 million each year.

Duke University, along with five other sites: San Francisco, Mississippi, Denver, Virginia Beach, and Charleston, is actively working to create solutions to the shortage of computer scientists with support from the National Science Foundation (NSF), Grant ESI-0624642 rant support from the National Science Foundation. As part of this grant, Professor Susan Rodger of Duke University is facilitating a workshop held at Duke during the summer of 2008 to provide middle- and high-school students with training to integrate the Alice programming language into the curricula offered in various North Carolina public schools.[6] Teachers will be trained for one week in Alice programming, and will then spend two weeks on developing curriculum materials. They will also have opportunities to test some of those materials on students attending a summer camp.

The goal for this project was to address declining interest in computer science by exposing students at the elementary school level to an animated Alice World that could be modified by teachers or students. This would serve the dual purpose of providing young students with computer programming basics and boosting reading comprehension since the animated Alice World would be devised to increase reading comprehension.

Alice has already been shown to have a positive effect on at-risk computer science students at the collegiate level;[7] and because of its animated features, it seems likely that younger students would also benefit from being introduced to Alice.

This project primarily involved designing a short program to provide primary and secondary school teachers with some techniques and tools for teaching programming with Alice to their students. The possibilities included a supplement to a math or history lesson. As the project got underway, however, it became apparent that a tool to facilitate reading comprehension would be the focus for this project.

Description of Alice and Prior Work

Alice is a young programming language, having been developed in 2002 by Professor Randy Pausch and his team at Carnegie Mellon University, and is available as a free download.[8] Although originally designed to be used in conjunction with another programming language, it has evolved into a tool that serves as a user-friendly introduction to the concepts that define computer science.[9]

Its popularity and success as an introductory learning tool is most likely attributable to Alice's drag-and-drop interface. One of the most frustrating things about learning how to program is discovering that your program doesn't work due to a syntax error, because they are so difficult to locate and the debugging process can be so very time-consuming. Since the syntax in Alice is in drop-down menus, the catalysts for frustration are greatly reduced.

Some of the concepts of object-oriented computer programming are difficult to grasp[1]. Since Alice uses 3-D virtual representations, it lends itself to visualization of complicated concepts; Alice helps make abstract concepts more palatable to novice programmers.[10]

As an educational tool, programs using animated Alice Worlds can be used in grammar school to enhance multiplication skills, for example, or to help teach a history lesson. There is also a version of Alice (Storytelling Alice[11]) that can be used to teach computer science to younger students. Alice animations can also be used to create short games or quizzes that aid reading comprehension, which is the way in which Alice (version 2.0) was utilized for this project.

In July 2007, I assisted Dr. Susan Rodger at a workshop[6] that she held at Duke University to introduce high school girls to the Alice programming language. By the end of the morning, every participant had followed the given directions and created an animated scene involving a couple of snowpeople. This workshop, because of the way in which Alice is structured, helped the girls learn about the distinctions between the concepts of methods and events (something which can take considerably longer in a "regular" beginning programming class), and recursion (see Appendix I). Prior to the July workshop, Dr. Rodger had taught a similar workshop for younger females (4th-6th grades), also with success, although the high school students were able to cover more material in the same amount of time as their younger counterparts.

There have been a number of investigations into what works and does not work in teaching computer science, some of which have been pioneered by the authors[12,13] of Learning to Program with Alice,[9,] as well as inquiries into the apparent decline in Computer Science among students,[14] despite the expected future demand for IT professionals. Professor Rodger and a host of other professionals have also reported extensively on ways to integrate algorithm animations into computer science courses.[15,16,17,18]

Interactive Alice World Quiz To Enhance Reading

We met with three administrators of the Durham Public School System to discuss the possibility of preparing educators to teach Alice to their students (focus on grades 4-8), and Professor Rodger's ideas were very well received. We met with Ron Wahlen, Executive Director of Instructional Technology, Janet Scott, Director of Science Grades 6-12, and Pam Sickles, Director of Match for Grades 6-12. Afterward, we met with Jan Riggsbee, Director of the Duke Elementary Teacher Preparation Program, and of the Program in Education (PiE),[19] to discuss the favorable reception of Alice by Durham public school administrators and to determine how to best incorporate Alice into the teacher training program.

The decision was made to focus on creating an Alice animation based on the popular Magic Tree House children's book series.[20] I obtained four of the books from the series, based on some of the 3-D objects I had seen in the Alice galleries, and decided to create an interactive game-like quiz based on the first book in the series, Dinosaurs Before Dark.[21] I wanted to create a game similar to 'Jeopardy' – it would be a board game where the players advance by answering questions correctly. Questions would be worth varying numbers of squares to move ahead. A wrong answer would move the player back a square. Before programming, the first task was to read the book and create a list of questions with their answers.

Another idea was to create a game in which the user could select one or two players, and they could each choose a pawn. The playing board environment would reflect various scenes from the books, such as an area that would include an oak tree and a tree house, resembling Frog Creek. Another section would be the interior of the tree house. Yet another area would have a hilltop looking down at a lake and the anatosaurus babies. Still another would show t-rex, etc. I also thought about juxtaposing colored shapes to create a path from start to finish.

Each correct answer would be worth 3 spaces along the path, and an incorrect response would move the pawn backward one space (with three tries total per question). Pressing the space bar would highlight the correct answer, without penalty. There would be no need to keep score, since winning would be obvious by reaching the end. A timer, however, might be necessary (treating the end of the timer the same way as pressing the space bar). Perhaps upon reaching the end of the path (i.e., winning the game), a flying reptile would pick up the pawn and fly him back to the starting position. The player would then be presenting with the option of playing again. After playing around with the program, I decided this was a little too ambitious a project, so I devised a more realistic animation. Instead of a board game, I decided that a multiple choice quiz would be the best way to go. Some of the roadblocks included how to display and increment or decrement the score.

I began to populate the AliceWorld with objects and quiz questions and created a method for scoring, then realized that I had forgotten to pass parameters. I incorporated nested loops to keep track of the iteration through the quiz questions and the number of guesses a player has used per question, then re-created a simple scoring system. Another problem involved how to display the instructions because the animated world would not stop when I pressed the key to show instructions. Repeated attempts at solving that problem by moving the instructions to a different view proved to be unsuccessful. Because I had already put in a lot of new information I didn’t want to go back to an earlier backup. Instead, I opened up an earlier backup and printed the code I had written and I reinserted it. It wasn’t perfect but at least I didn’t have to recreate the proverbial wheel.

I also wanted to create a different animation for each correct answer, but in the interest of time, I kept it simple and repeated the correct answer animation. As teachers become more comfortable with Alice, they can easily add more animations to DinoQuiz. I was finally satisfied with how the introduction worked but I was not certain about how to display the questions. If I continued as I did for question 1, the game would continue without waiting for user input. A similar problem was solved earlier by having the user input yes or no. That method was impractical for each possible choice of answer because it would have been very cumbersome and sloppy. My preference was to have a list (array) of correct answers and then iterate through to see if they were correct. (I mostly did the passing of parameters to set the game up according to whether Annie or Jack was clicked on.) However, as with the clicking of Jack or Annie, in no click occurred, the game continued anyway. Using Alice helped to clarify some ambiguities in my understanding of passing parameters and sharing them within various methods. I then worked on creating the arrays for the questions and answers. There was some confusion involving my backups which necessitated my creating new codes for both DinoQuiz and DinoQuiz New Format. I went back to the earlier version DinoQuiz because it had everything that I needed. I completed mot methods and arrays. I then had to figure out how to iterate through the questions and call the correct methods without causing errors.

The completed DinoQuiz[22] proceeds as follows (Screen shots are available in Appendix II):

The first screen introduces the quiz and instructs the user to either click on "next" or type "x" to skip the instructions and select a pawn. The instructions explain how to select a response to the questions, and outlines the scoring system. Some of the text changes color for emphasis, before proceeding to the screen for selecting the pawn. When the user clicks on an object (boy or girl), the chosen pawn displays a short animation before progressing to the quiz. Each correct answer earns two points, whereas the penalty for an incorrect or "I don't know" response is one point (these can be changed by the teacher, as proficiency is gained using Alice); the lowest possible score is zero (no negative points). Currently, there are 13 questions loaded into DinoQuiz; however, adding more is a simple feat. Both an "I don't know" and an incorrect response highlight the correct answer in an effort to help boost comprehension of the book.

After the last question has been responded to, the user is prompted with a question asking whether he or she wishes to view the game statistics (total questions, total points, number of questions answered correctly, incorrectly, or with an "I don't know" response). The program will proceed to the statistics screen, regardless of the user's response.

Conclusion and Future work

Learning basic programming strengthens critical thinking skills, and Alice, which is a graphical front-end for Java, virtually eliminates syntax errors, which are generally frustrating for everyone (even experienced and seasoned programmers, because they are so difficult to spot) provides an innovative tool for teaching basic computer science skills to young students. The ALICE process allows for demonstration of the basics of design and debugging in a novice-friendly environment that automatically minimizes syntax mistakes. My goal was to design and develop a children's game in ALICE that could be utilized by a targeted elementary school (4th grade) teacher to reinforce reading comprehension. The premise is that the teacher could easily modify the program to match specific classroom needs, and can simultaneously be used as a hands-on tool for the students to experiment with programming.

Incorporating the use of Alice in grade school will help encourage young students to pursue computer science. At the very least, it will help build the critical thinking skills that are so necessary for success.

It would be nice to have future work include conducting a study of whether reading comprehension is actually improved in students that play the game. Perhaps quiz them on a book from the series and compare those results to the same students who take the animated quiz. Also, there can be two groups of students that have similar test-taking result histories, and have one group take the quiz created with Alice while the other group has the same questions on a paper quiz.

Future enhancements to DinoQuiz could incoude:

- Making the initial value of each question 3 points, and giving the user two re-tries for fewer points each time;
- Offering the correct response without penalty for an "I don't know" response;
- Adding a timer that defaults to "I don't know" if the timer runs out;
- Add more animations for correct and incorrect responses;
- Add a "Quiz Over" animation, with the option to try again;

- As part of the statistics, add a counter for number of questions answered correctly on the first, second, and third tries, independently of each other.
- Creating a database of 100 or so questions that load randomly each time DinoQuiz is operated.

Also, teachers can add questions, and additional animations, or can have children do so in order to build and strengthen computer science and related skills, such as critical thinking.

Acknowledgements

Funding for this internship was provided by the Computing Research Association Distributed Mentor Project (CRA-W DMP). The objective of CRA-W DMP is to increase the number of women entering graduate studies in the fields of computer science and engineering. This highly selective program matches promising undergraduate women with a faculty mentor for a summer research experience at the faculty member's home institution. Students are directly involved in a research project and interact with graduate students and professors on a daily basis. This experience is invaluable for students who are considering graduate school, providing them with a close-up view of what graduate school is really like and also increasing their competitiveness as an applicant for graduate admissions and fellowships.

Susan H. Rodger, Ph.D., my CRA-DMP mentor at Duke University, who was patient with me while I suffered through the learning curve and whom I hold in exceptionally high esteem.

Beth Simon, my first CRA mentor (and my first programming professor at UCSD) who was also a co-mentor for the second part of this project, when I had to complete the project in San Diego. She has always kept her door open for me and imparts valuable, wise advice.

Peer review of draft reports, help with testing and debugging, and for generally being available to hear my crazy ideas or to just listen in general: Trisha Mejilla, Michael Rogers, Ryan Hertzog, John Torres, Krista Davis, Paloma Dominguez, and Gil Dominguez.

Literature Cited

[1] Vegso, Jay. Interest in CS as a Major Drops Among Incoming Freshmen. May 2005. Computing Research Association. http://www.cra.org/CRN/articles/may05/vegso.

[2] Slater, Don. 3D Animation: Programming though storytelling. Carnegie Mellon University, 2006. http://www.cs.cmu.edu/~djslater/CS4HS_Alice_Intro.ppt.

[3] Davy, J. R., Audin, K., Barkham, M., and Joyner, C. 2000. Student well-being in a computing department. In *Proceedings of the 5th Annual SIGCSE/SIGCUE Iticse conference on innovation and Technology in Computer Science Education* (Helsinki, Finland, July 11 - 13, 2000). ITiCSE '00. ACM, New York, NY, 136-139. http://doi.acm.org/10.1145/343048.343145.

[4] Hagan, D. and Markham, S. 2000. Does it help to have some programming experience before beginning a computing degree program? *SIGCSE Bulletin* 32, 3 (Sep. 2000), 25-28. http://doi.acm.org/10.1145/353519.343063.

[5] National Assessment of Adult Literacy. Survey by the National Center for Education Statistics. U.S. Department of Education, 2003.

[6] Rodger, S.H. Adventures in Alice Programming. Duke University, 2007. http://www.cs.duke.edu/csed/alice/dukeFemmes/.

[7] Moskal, B., D. Lurie, and S. Cooper, Evaluating the Effectiveness of a New Instructional Approach. SIGCSE 2004, Norfolk, Virginia. March 2004.

[8] Carnegie Mellon University. Alice website. http://www.alice.org/index.php?page=downloads/download_alice.

[9] Dann, Wanda P., Stephen Cooper, and Randy Pausch, Learning to Program with Alice, Pearson/Prentice Hall, Upper Saddle River, NJ. 2006.

[10] Carnegie Mellon Collaborates with EA to Revolutionize and Reinvigorate Computer Science Education in the US. Press release from Alice. March 10, 1996. http://www.alice.org/index.php?page=sims_announcement/sims_announcement.

[11] Kelleher, Caitlin. Animations for Storytelling. http://www.cse.wustl.edu/~ckelleher/research.htm.

[12] Cooper, Steven, Kenneth J. Goldman, Martin Carlisle, Myles McNally, and Viera Proulx, Tools for Teaching Introductory Programming: What Works?, presented at SIGCSE 2006.

[13] Cooper, Steven, Wanda Dann, and Randy Pausch, Teaching objects-first in introductory computer science, presented at SIGCSE 2003, Reno, Nevada. February 2003.

[14] Rodger, Susan H. An Innovative Approach with Alice for Attracting K-12 Students to Computing. International Conference on the Virtual Computing Initiative (IBM University Days), Research Triangle Park, NC, May 7, 2007, (p. 17).

[15] Rodger, S.H. Integrating Animations into Courses, ACM SIGCSE/SIGCUE Conference on Integrating Technology in Computer Science Education, Barcelona, Spain, p. 72-74, 1996.

[16] Rodger, S.H., Introducing Computer Science Through Animation and Virtual Worlds, Thirty-third SIGCSE Technical Symposium on Computer Science Education, p. 186-190, 2002.

[17] Cooper, Steven, Wanda Dann, and Randy Pausch. Developing Algorithmic Thinking With Alice. Presented at ISECON, Philadelphia, Pennsylvania. November 2000.

[18] Cooper, Steven, Wanda Dann, and Randy Pausch, Alice: A 3-D Tool for Introductory Programming Concepts, presented at CCSNE, Ramapo, New Jersey. April 2000.

[19] Program in Education (PiE), Duke University. http://fds.duke.edu/db/aas/Education/faculty/jrigg.

[20] Osborne, Mary Pope. The Magic Tree House. http://www.randomhouse.com/kids/magictreehouse/series.html.

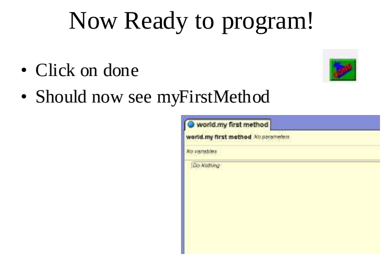
[21] Osborne, Mary Pope. Dinosaurs Before Dark. Random House. http://www.randomhouse.com/kids/magictreehouse/dinosaurs.html.

[22] Lopez, Mercedes internship website. Page offers a link to DinoQuiz (scroll down to the bottom of the web page) http://www.cs.duke.edu/~mrl19/the_project.html.

Appendix I

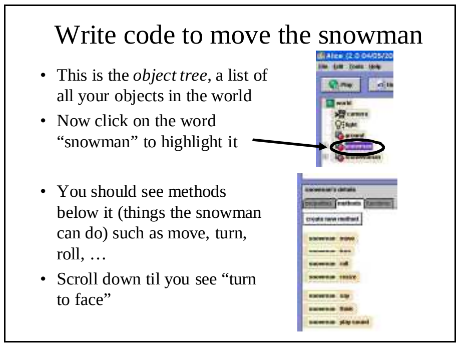
Excerpted instructions from Professor Rodger's workshop.

Figure 1



Now Ready to program!

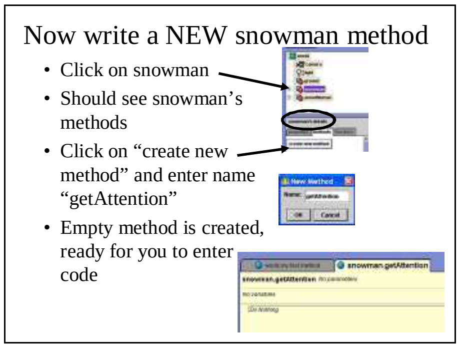
- Click on done
- Should now see myFirstMethod



Write code to move the snowman

- This is the *object tree*, a list of all your objects in the world
- Now click on the word "snowman" to highlight it
- You should see methods below it (things the snowman can do) such as move, turn, roll, …
- Scroll down til you see "turn to face"

Figure 2

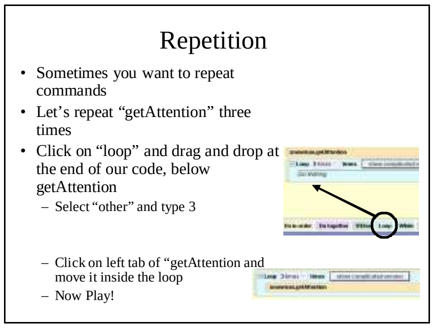


Now write a NEW snowman method

- Click on snowman
- Should see snowman's methods
- Click on "create new method" and enter name "getAttention"
- Empty method is created, ready for you to enter code

Figure 3

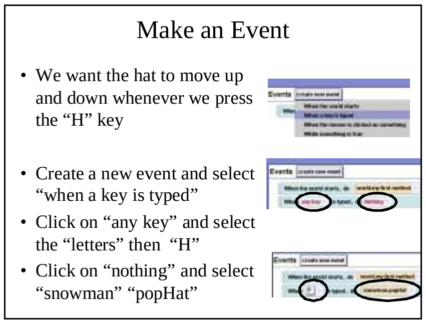
Figure 1 prepares the participants for creating their first method for the objects they just created for their animated scene using Alice. Figures 2 and 3 help to describe how to use methods. Figures 5 and 6 provide help with events.



Repetition

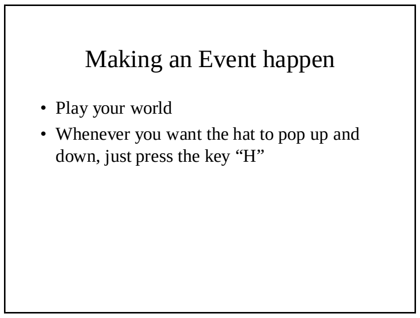
- Sometimes you want to repeat commands
- Let's repeat "getAttention" three times
- Click on "loop" and drag and drop at the end of our code, below getAttention
  - Select "other" and type 3
  - Click on left tab of "getAttention and move it inside the loop
  - Now Play!

Figure 4 describes the concept of recursion.



Make an Event

- We want the hat to move up and down whenever we press the "H" key
- Create a new event and select "when a key is typed"
- Click on "any key" and select the "letters" then "H"
- Click on "nothing" and select "snowman" "popHat"



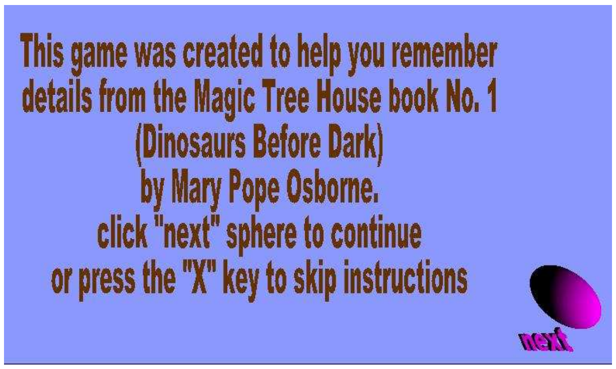
Making an Event happen

- Play your world
- Whenever you want the hat to pop up and down, just press the key "H"

Figure 5 **Figure 6**

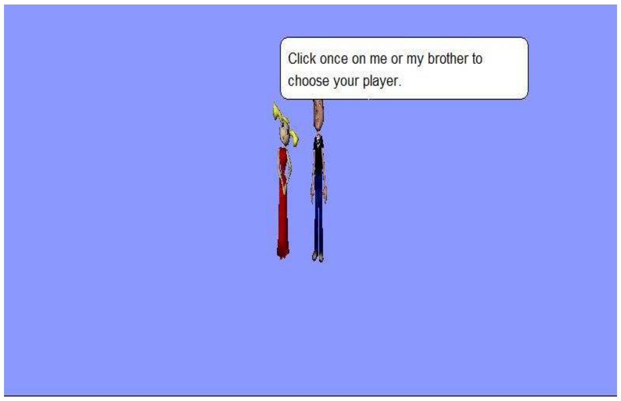
Appendix II

Samples from DinoQuiz



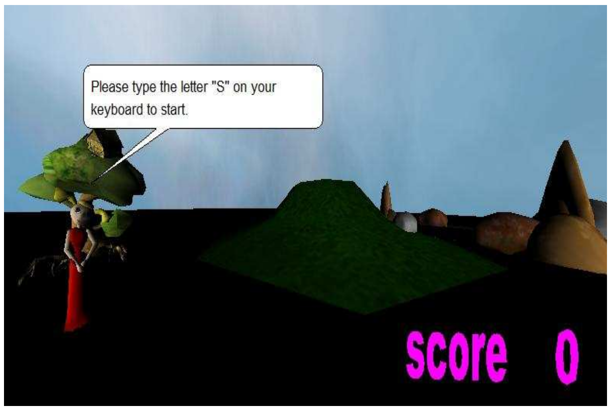
This game was created to help you remember details from the Magic Tree House book No. 1 (Dinosaurs Before Dark) by Mary Pope Osborne. click "next" sphere to continue or press the "X" key to skip instructions

Figure 7: This is the opening screen that displays the introduction. The user clicks on the "next" sphere to view the instructions, or types an "x" to skip the instructions and select a player token.



Click once on me or my brother to choose your player.

Figure 8: This is the image that is displayed if the instructions are skipped (by typing “X” from the initial screen). First the girl is instructing the user to click once on an image to select the game character. If the girl is selected, the boy does a disappearing animated sequence and the score will be displayed in magenta. If the boy is selected, the girls does the disappearing animation and the color of the score display will be orange.



Please type the letter "S" on your keyboard to start.

Figure 9: The characters represent the main characters from the award-winning book series, The Magic Tree House by Mary Pope Osborne. Here, the user clicked on the female icon, and the screen changed to the initial game screen where the character tells the user to press the "s" key to start the questioning.

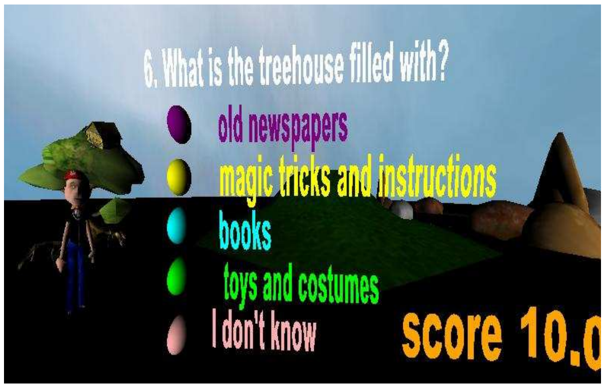

6. What is the treehouse filled with?
old newspapers
magic tricks and instructions
books
toys and costumes
I don't know
score 10.0

Figure 10

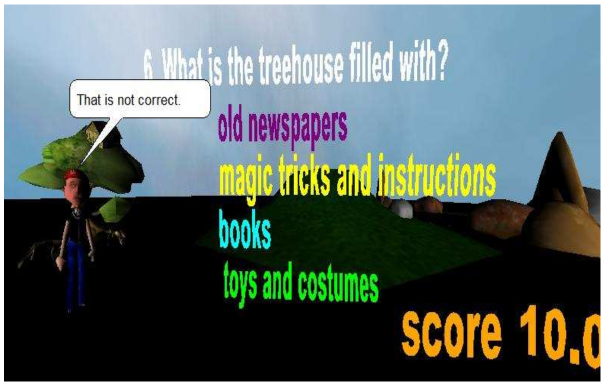

6. What is the treehouse filled with?
That is not correct.
old newspapers
magic tricks and instructions
books
toys and costumes
score 10.0

Figure 11

Figures 10 and 11: Here's a sample question, in which the user selects an incorrect response. In Figure 4, the program is waiting for the user to click on a sphere indicating the user's response. The spheres disappear after the user makes a selection. Figure 5 depicts the character's face turning red while he tells the player that he or she selected an incorrect answer. The correct answer will also be highlighted with an animation.

Figures 12 and 13: The next two images illustrate what happens if the user clicks on the sphere next to "I don't know". First, the character says, "I will show you the correct response". Then, the correct response changes color while pivoting in a circle.

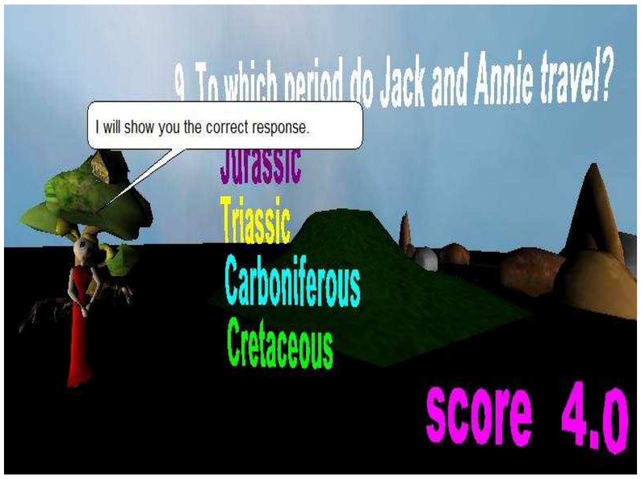

9. To which period do Jack and Annie travel?
I will show you the correct response.
Jurassic
Triassic
Carboniferous
Cretaceous
score 4.0

Figure 12

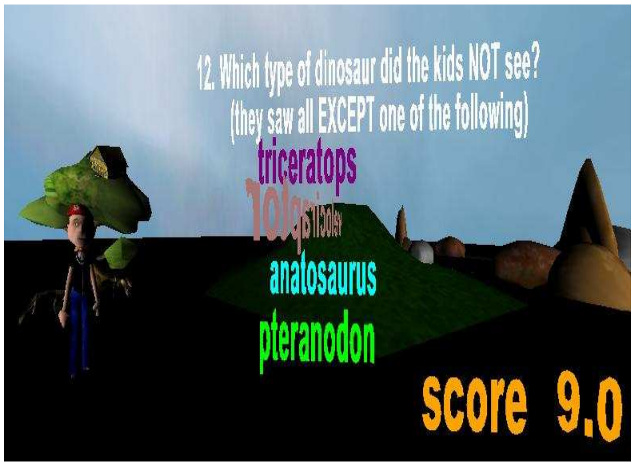
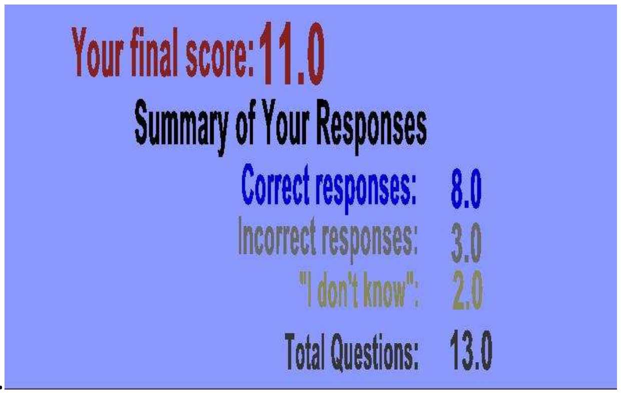

12. Which type of dinosaur did the kids NOT see?
(they saw all EXCEPT one of the following)
triceratops
anatosaurus
pteranodon
score 9.0

Figure 13



Your final score: 11.0

Summary of Your Responses

Correct responses:	8.0
Incorrect responses:	3.0
"I don't know":	2.0
Total Questions:	13.0

. **Figure 14:** When the game ends, the player is redirected to a screen displaying the game statistics. Information is not saved, so the information is only for the game most recently completed.