To be presented at AERA 2011 Symposium: Merging Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom (Repenning et al).

# Measuring Computational Thinking in Middle School using Game Programming

Jill Denner & Linda Werner
ETR Associates, Scotts Valley, CA and UC Santa Cruz, Santa Cruz, CA

## Objectives.

Computational thinking (CT) involves using the techniques of computer science to systematically and efficiently solve problems and process information, and there is growing interest in incorporating CT into K-12 (e.g., recent efforts by the Computer Science Teachers Association, the International Society for Technology in Education, and Google). However, the construct of CT does not lend itself well to questionnaire assessments; it requires more qualitative, in-depth measures. In this paper, we will describe one strategy for measuring CT among middle school students enrolled in a game programming class. The fairy assessment is designed to measure students' understanding of the programming environment, as well as their understanding of aspects of CT such as abstraction/modeling and scale, and whether they can apply algorithmic thinking to solve a problem. The findings are being used to describe what computational thinking looks like in middle school, and to test and refine the measure.

## Background

Computational thinking (CT) was first described by Seymour Papert (1993) and then pioneered by Jeannette Wing in 2006. "Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2008). Research has shown that programming games is highly motivating for students and involves certain aspects of CT (Ioannidou, Repenning, & Webb, 2009; Werner et al., 2009). But assessment remains a challenge. Most strategies used to measure CT among child programmers use their final products (e.g., games or models) as indications of their higher level thinking (Denner, Werner, & Ortiz, under review; Repenning et al., 2010). But other research suggests that examining the process of debugging-- whether presented with a buggy task or dealing with one's own bugs—is a useful strategy for measuring complex reasoning (Games, 2010; Ioannidou et all, 2009; Kurland et al., 1989). The fairy assessment is designed to measure different aspects of CT by engaging students in a high level narrative and fixing the code that runs it.

## Methods and Data Sources.

The classes met for between 25-30 hours and students programmed a game using the programming environment Alice or Storytelling Alice (Dann, Pausch, & Cooper, 2006; Kelleher & Pausch, 2007). At the end of the course, students were presented with an interactive program about fairies written in Alice, and asked to play it and then follow the directions to modify it. Some directions were given on paper; others were communicated via game character dialog. *Task One* was to make the female fairy watch the male fairy while he is walking into a forbidden forest. *Task Two* was to help the male fairy return to his original size because a tree in the forbidden forest has shrunk him. The *Third task* was to help the male fairy fly out of the forbidden forest to the female fairy when the user clicks the mouse on him. Student performance on each task was coded on a scale from 0 to 10, with a maximum assessment score of 30.

Data from 138 students has been coded so far. These students attend six different schools with variation across race/ethnicity (primarily White and Hispanic/Latino) and socioeconomic status.

## Results

The mean scores for the Fairy Assessment are in Table 1. On average, students scored highest on Task One, and lowest on Task Two, although there was a range of complexity in their solutions. For example, the simple solution for Task One would be a value of 4, which indicates that a student can read the code and understand enough about the algorithm used and about abstraction and modeling to determine where to place any one of the numerous Alice programming instructions that can cause the female fairy to face the male fairy after he has completed his walk into the forbidden forest. A more complex solution to Task One would be a value of 10. Students that scored 10 understand the concepts of parallelism (one fairy watches while the other fairy is walking) and abstraction and modeling (passing a value to a method (procedure) to set the length of time the method executes).

A simple partial solution for Task Two (a value of 6) indicates that students understand how abstraction and modeling work in Alice and can identify what part of the code needs to be fixed or replaced and then fixes or replaces it. In this case, it is a method call to return the fairy to its original size. If the student is able to fix it, this shows an understanding of one concept of algorithmic thinking called conditional execution—how to alter the sequential execution of a program using conditional logic. If the student replaces the method correctly, this shows that the student understands abstraction and modeling enough to connect an existing mouse click event to a different method to alter the size of the fairy. A more complex solution for Task Two (coded as a 10) suggests a deeper understanding of algorithmic thinking because no automatic changes in the size of the fairy occur elsewhere in the program and the event is appropriately connected to a correctly executing resizing method.

Students who create a simple solution for Task Three (coded as a 4) have demonstrated an understanding of a fairly complex concept about algorithmic thinking--that events alter the sequential execution of a program (e.g., something happens when the player clicks the mouse on an object). A more complex solution for Task Three (coded as a 10) shows an understanding of a concept of algorithmic thinking and abstraction and modeling because a method chosen (to fly) is connected to the mouse click event and values are passed to the method to indicate how many times to fly at each mouse click and to whom to fly toward.

Table 1.
Mean scores on Fairy Assessment

| Task | Average Percent Correct | Range of Percent Correct |
|---|---|---|
| 1 | 6.21 | 0-10 |
| 2 | 4.14 | 0-10 |
| 3 | 5.78 | 0-10 |

## Discussion

The performance assessment was designed to be motivating and engaging to middle school students--the use of a game-like format and animated (and talking) fairy and tree objects created a familiar environment to test higher order thinking. Each one of the three tasks required the student to identify and integrate various solution pieces, and then make these pieces work

together with the existing program. In order for students to complete the assessment, it was necessary to first understand the story narrative, and the scores indicate that most did. Differences in scoring were due to different demands on students' reasoning. For example, Tasks One and Three required students to add new functionality to the program, whereas Task Two required students to fix something in the program code. In all three tasks, students needed to use algorithmic thinking and abstraction and modeling knowledge, however, debugging is more difficult because the fault in the algorithm must be found before a solution approach is designed and then implemented. For Tasks One and Three, students did not need to find faults in the algorithms that were implemented by the code; they only had to design solution approaches and implement them.

A closer example of the details of each of the tasks shows that the methods and values passed to the methods (via parameters) used to solve Task One were built-into Alice and all students were very familiar with them. These methods and parameters are available for almost all of Alice's built-in objects. The methods and their parameters involved with Tasks Two and Three were created specifically for the assessment and were unfamiliar to the students even though the students had received instruction in the use and creation of user-defined methods. The method involved with Task Three required the assignment of values for two parameters. The use of two parameters was novel for many students because few user-defined methods have more than one parameter and many have none. Task Two also involved understanding conditional logic, a difficult aspect of algorithmic thinking.

The assessment provides data on whether students can apply CT concepts they learned in the class to a new situation. While this assessment approach provides some insight into certain aspects of computational thinking in middle school, limitations are that it can only be used after students gain familiarity with Alice, and thus can only be used to measure changes in CT over time once students become proficient with the tool. While the Fairy Assessment itself may not be generalizable to other programming environments, the gamelike approach can be used in other settings. We suggest that each task should be designed to measure certain aspects of CT (e.g., in our case we measured algorithmic thinking and abstraction and modeling) and each task should be as independent as possible from the other assessment tasks. While assessments like these are not good measures of how CT develops, they provide a window into what CT looks like in middle school, including the range of thinking, and the areas that are more or less challenging for this age group.

## References


Dann, W.P., Cooper, S., & Pausch, R. (2006). *Learning to Program with Alice*. Pearson/Prentice Hall.

Denner, J., Werner, L., & Ortiz, E. (under review). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education.*

Games, A. (2010). Bug or feature: The role of Gamestar Mechanic's material dialog on the metacognitive game design strategies of players. *E-Learning and Digital Media, 7(1),* 49-66.

Ioannidou, A., Repenning, A., and Webb, D. C. 2009. AgentCubes: Incremental 3D end-user development. *Journal of Visual Language and Computing,* volume 20, issue 4, doi: 10,1016/j.jvlc.2009.04.001. 236-251.

Kelleher, C. & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM, 50(7),* 59-64.

Kurland, D.M., Pea, R.D., Clement, C., & Mawby, R. (1989). The study of the development of programming ability and thinking skills in high school students. In E. Soloway & J.C. Spohrer (Eds.), *Studying the novice programmer*, pp. 83-112. Hillsdale, NJ: Erlbaum.

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas.* Cambridge, MA: Perseus Publishing.

Repenning, A., Webb, D., and Ioannidou, A. 2010. Scalable game design and the development of a checklist for getting computational thinking into public schools. *SIGCSE '10.* (March 2010), 265-269.

Werner, L., Denner, J., Bliesner, M., Rex, P. 2009. Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study. *Proceedings of the 4th International Conference on Foundations of Digital Games.* 207-214.

Wing, J. 2006. Computational thinking. *CACM* vol. 49, no, 3. March 2006, pp. 33-36.

Wing, J. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society*, vol. 366, July 2008, pp. 3717-3725.