Promoting Computational Thinking in children Using Apps

Ms. Hoda Ehsan, Purdue University, West Lafayette (College of Engineering)

Hoda is a Ph.D. student in the School of Engineering Education, Purdue. She received her B.S. in mechanical engineering in Iran, and obtained her M.S. in Childhood Education and New York teaching certification from City College of New York (CUNY-CCNY). She is now a graduate research assistant on STEM+C project. Her research interests include designing informal setting for engineering learning, and promoting engineering thinking in differently abled students in informal and formal settings.

Ms. Chanel Beebe, Purdue University, West Lafayette (College of Engineering)

Chanel Beebe is an Engineering Education Researcher at Purdue University where her work focusing on broadening participation in engineering and engineering thinking. Her passion lies in empowering communities to solve their own problems using creative pedagogies and engagement strategies.

Dr. Monica E Cardella, Purdue University, West Lafayette (College of Engineering)

Monica E. Cardella is the Director of the INSPIRE Institute for Pre-College Engineering Education and is an Associate Professor of Engineering Education at Purdue University.
Promoting Computational Thinking Using Apps

Introduction

Researchers within the field of Engineering Education are looking for more ways to incorporate engineering and engineering thinking into both K-12 formal and informal setting. Increasing demand for curricula and programming that supports computational thinking in K-2 settings motivates our research team to investigate how computational thinking can be understood, observed, and supported for this age group. One way to integrate computational thinking in K-2 education is with the use of educational apps. We used our preliminary understanding of computational thinking to develop our guiding codebook. The codebook includes INSPIRE definition, and examples and non-examples observed in the apps. Through a systematic approach, we reviewed 89 apps and finally identified 12 educational app that promote computational thinking in the context of problem-solving. The apps and the computational thinking competencies that each app promotes are listed in this study. For the field of engineering education at large, the results of this study illuminate the following points:

1. Computational thinking is possible to observe and teach at the K-2 levels.
2. Educational media, especially apps, can be used to promote computational thinking competencies.

The codebook can serve as a tool to review other educational media that promote computational thinking. In addition, the apps identified in this study can be integrated into both formal and informal learning activities. The next studies include reviewing and identifying more apps, reviewing books and games, and observing children playing with the apps and games to investigate what computational thinking competencies look like in children.
Background

Incorporating engineering and engineering thinking into both K-12 formal and informal settings has gained increased attention from engineering education researchers. More recently, incorporating Computational Thinking (CT) has gained increased attention in both formal and informal settings. Therefore, many educational resources have focused on integrating engineering and computational thinking with other subjects in their standards and curricula (e.g. CSTA, 2012, NGSS 2012). As an example, the Next Generation Science Standards include engineering and computational thinking aligned with the science practices. Additionally, Dasgupta and Purzer (2016) stated that if CT competencies are integrated into STEM education, it can positively impact STEM learning for K-12 students. Therefore, understanding what computational thinking competencies look like in K-12 education is useful for the future of STEM Education.

In 2011, the National Research Council Report of a Workshop of Pedagogical Aspects of Computations characterized engineering as a key focus of computational thinking in elementary education. In this report, Cunningham connected computational thinking and engineering problems to assert that computational thinking was crucial to engineering habits of mind. These habits of mind describe how values, attitudes, and thinking skills are linked to engineering. Computational thinking has also previously been linked to engineering beyond simply programming by Wing in 2006. Wing defined computational thinking as the overlap between mathematical thinking and engineering thinking.

In 2011, The Computational Thinking Teacher Resources developed as the result of a collaboration between the Computer Science Teachers Association (CSTA) and the International Society for Technology and Education (ISTE). This collaboration produced a list of characteristics that define and describe computational thinking and its qualities. In 2012, Google also released a list of computational thinking competencies and they provide teaching tips on how to elicit this type of learning in students. However, the competencies identified by Google do not entirely match those presented in the Computational Thinking Teacher Resources, and differences in how computational thinking is defined and operationalized also varies across the research literature.

Purpose of the study

As a part of a larger study, we are interested in observing computational thinking in museum exhibits and homeschool settings, as well as understanding what tools promote computational thinking within these spaces. These potential tools include, but are not limited to, digital media and games. As a result, our team has aimed to review digital media, books, and games that claim to promote the development of computational thinking in children.
Digital media is taking over adults’ lives and children’s of all ages. Digital media such as tablets and applications have entered the educational system and become popular worldwide. Chiong and Shuler (2010) demonstrated that apps could successfully sustain children’s’ learning as well as their interest. Moreover, Couse and Chen (2010) called for “more fully integrate technology into the curriculum to encourage the active engagement and thinking of young children” (p.76). Hence, in order to integrate apps into educational material and curriculum, it is important to select the apps that promote children’s learning. Therefore, the goal of this study is to review and select the apps that potentially promote computational competencies in K-2 children.

**Preliminary Understanding of CT**

To understand how computational thinking may be incorporated into K-2 settings, our research team used the computational thinking competencies described by CSTA and Google as our initial framework. We choose these models as they included breakdowns of CT competencies for our target group.

Using these preliminary definitions, our research team reviewed videos of K-2 students engaged in PictureSTEM curricular units (Hynes et al., 2016). PictureSTEM is an integrated STEM+literacy curriculum developed for grades K-2 with the focus on engineering design and literacy (Tank, Moore, & Pettis, 2013). The goal of reviewing these videos was to see what computational thinking looks like when children engaged in engineering activities. Through this review, we highlighted student interactions that seemed to be examples of CT. Figure 1 shows an example of what we have found.

**Table 1. Review Book**

<table>
<thead>
<tr>
<th>Clip Number: Lesson 1B-2 literacy discussion</th>
<th>Student Activity: experimenting with using different objects as measurement tools.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competency</td>
<td>Time: 6.00-8.00 13.30-17.00 15.00-17.01 16.33-18.00</td>
</tr>
</tbody>
</table>
### Data analysis

**Measuring the toy box together**
Explain their units and making sense of patterns about the size of the unit. The girl: “Some units are bigger.”

**The little girl describes her work:** “We just have a different units! Box were not different.” This object is bigger than this object, it gets more about of we are measuring. Therefore this one is obviously bigger than this one.

**The little girl realizes that the card she was measuring with has two different sizes to measure so it gives her two different measures.**

### Data representation

**Teacher represented data in a chart.**

**Abstraction**

**Making sense of what Standard is.**

### Simulation

The girl shows the card to her friends, and illustrates how using this card and its two different sides gives them two different answer.

Reviewing this videos provided evidence that students in this age group are able to engage in different computational thinking competencies. Also, the examples provided us with enough insights to synthesize, develop and finalize our CT definitions (INSPIRE Definitions, 2017). To compare the definitions, Appendix 2 presents three sets of CT definitions including INSPIRE, Google and CSTA.
Reviewing Educational Apps

We used the INSPIRE CT definitions as a framework to review and select educational media that engage children in those competencies. The educational media includes apps, games, toys, books, and websites. In this study, however, we focused on selecting and reviewing educational apps. First, we identified a sample of apps to review, developed a coding procedure, then coded selected apps for evidence of computational thinking. The following section describes these steps in detail.

Identifying a sample

To select our samples, we looked for the websites that meet four criteria: (1) offer age-based libraries, (2) include experts’ reviews and ratings, (3) include the parents’ reviews and ratings, and (4) have a search engine within the website. Among the websites we found that www.commonsensemedia.com met all four criteria. To find the apps that claimed to promote computational thinking, we used “computing”, “programing”, or “computational thinking” as our research keywords. We then used the inclusion criteria below to select the apps:

1) The app should be appropriate for ages 5-8
2) The app should be rated 3.5 or higher by the customers or parents based on the rating scheme below:
   Is it any good?
   Just fine (3 stars)
   Really good (4 stars)
   The best (5 stars)
3) The app should be rated 4 or higher by experts based on the criteria below:
   4 dots: Engaging, very good learning approach.
   5 dots: Really engaging, excellent learning approach.

Based on this criteria, we identified 86 apps to review.

Review procedure

The aim of the review process was to first exclude the apps that do not potentially promote computational thinking in children, and then to organize the remaining apps based on price, customer review and suggested age. To do so, we first created an initial review book which constitutes of age range, the price, platform, the rate given by costumers, the link for download/install, the potential computational thinking competencies, and a section to describe the potential competencies (see figure 3).
Next, we reviewed the apps by spending some time interacting with each app. In order to make decisions for excluding or including apps, we set a 10-minute critical time limit. We excluded the apps if no evidence of computational thinking competencies were found in them after 10 minutes. In average, the amount of time we spent on each app was between 10 to 30 minutes. Finally, this portion of the review process left us with 41 apps, each with at least one potential competency evident. These apps were then organized based on the other elements in the review book.

**Coding procedure**

Computational thinking is a process of problem-solving. Therefore, in the next phase of the study, we looked at computational thinking competencies through problem-solving lens using the set of definitions developed by our research team as a result of the video observations. The aims of the coding phase were to identify:

- A list of apps which develop computational thinking competencies in the context of problem solving in children.
- The computational thinking competencies which most frequently appeared in educational apps appropriate for K-2 aged children.

Each of the two researchers engaged in this process first coded one app individually. Next, we shared our experiences and findings to come into agreement about what certain activities in the apps required users to do. We then were able to generate examples and non-examples of
computational thinking. As we developed a collaborative understanding, we modified the codebook with examples and non-examples reflected in Appendix 2.

Next we used the codebook from Appendix 2 to code all 41 apps. Researchers spent exactly 30 minutes playing with each app and then another 30 minutes reflecting on their experience using the guiding codebook (Appendix 2) to identify existing competencies within the app.

**Results**

After coding the apps, we found that only 12 apps developed computational thinking through problem-solving. These apps and the competencies present in them are listed in Table 2. Abstraction, algorithm, pattern recognition, troubleshooting, and simulation were seen the most frequently in these apps. We did not see any evidence of data collection, data analysis, and parallelization in these apps.

**Table 2. Competencies Present in Apps**

<table>
<thead>
<tr>
<th>Name</th>
<th>Platforms</th>
<th>Price</th>
<th>Age</th>
<th>CT Competencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cargo-bot</td>
<td>iOS</td>
<td>Free</td>
<td>5+</td>
<td>Abstraction, Algorithms and Procedures, Debugging/Troubleshooting, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Codeable Crafts</td>
<td>Android</td>
<td>Free</td>
<td>5+</td>
<td>Abstraction, Algorithms and Procedures, Debugging/Troubleshooting, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Daisy the Dinosaur</td>
<td>iOS</td>
<td>Free</td>
<td>7+</td>
<td>Abstraction, Algorithms and Procedures, Debugging/Troubleshooting, Simulations, Pattern Recognition, Problem Decomposition</td>
</tr>
<tr>
<td>Kodable</td>
<td>iOS</td>
<td>Free</td>
<td>5-8</td>
<td>Abstraction, Algorithms and Procedures, Automation, Data Representation, Debugging/Troubleshooting, Simulations,</td>
</tr>
<tr>
<td>App</td>
<td>Platform</td>
<td>Availability</td>
<td>Age Range</td>
<td>Curriculum Features</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Lightbot Jr.</td>
<td>Web browsers, iOS, Android</td>
<td>Free</td>
<td>4-8</td>
<td>Abstraction, Algorithms and Procedures, Data Representation, Debugging/Troubleshooting, Simulation, Pattern Recognition</td>
</tr>
<tr>
<td>PBS KIDS ScratchJr</td>
<td>iPad, Android</td>
<td>Free</td>
<td>6+</td>
<td>Abstraction, Algorithms and Procedures, Debugging/Troubleshooting, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Robozzle</td>
<td>Web browsers, iOS, Android, Windows phone</td>
<td>Free</td>
<td>6+</td>
<td>Abstraction, Algorithms and Procedures, Automation, Debugging/Troubleshooting, Problem Decomposition, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Run Marco!</td>
<td>Web browsers, iOS, Android</td>
<td>Free</td>
<td>6+</td>
<td>Abstraction, Algorithms and Procedures, Data Representation, Debugging/Troubleshooting, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Scratch Jr.</td>
<td>iPad, Android</td>
<td>Free</td>
<td>6+</td>
<td>Abstraction, Algorithms and Procedures, Data Representation, Debugging/Troubleshooting, Simulations, Pattern Recognition</td>
</tr>
<tr>
<td>Sushi Monsters</td>
<td>iOS</td>
<td>Free</td>
<td>4+</td>
<td>Problem Decomposition</td>
</tr>
<tr>
<td>The Foos</td>
<td>iOS, Android, Windows Phone, Kindle</td>
<td>Free</td>
<td>5+</td>
<td>Abstraction, Algorithms and Procedures, Data Representation, Debugging/Troubleshooting</td>
</tr>
</tbody>
</table>
Conclusions and Implications

In this study we aimed to select and review the apps that potentially promote computational thinking competencies in K-2 children. The codebook we used included the INSPIRE definitions and examples and non-examples of the competencies. We finally identified 12 apps that promote CT in the context of problem-solving. The findings of this study indicate that the apps can develop all CT competencies listed in the INSPIRE index with the exception of problem decomposition, data representation and parallelization (for these three competencies, we do not have evidence that they cannot be developed in an app – we just did not find evidence confirming that apps can help children develop these competencies).

The apps identified in this study can be used by teachers and parents who are interested in engaging their children in experiences that develop computational thinking. For researchers looking to instill more computational thinking in educational media, the lack of evident parallelization, data collection and analysis competencies points to the opportunity to develop activities that better promote these competencies. For the field of engineering education at large, the results of this study illuminate the following points:

- Computational thinking is possible to observe and teach at the K-2 levels.
- Educational media, particularly apps, can be used to promote computational thinking competencies.
- The guiding codebook developed in this study can be used to review educational media for computational thinking.

Future Studies

To expand the findings of this study, further research could include reviewing apps from other sources. In addition, future studies should include investigating what computational thinking looks like when children play with these apps, and what competencies can be observed.
developing in these children. Beyond educational applications, further phases of this research will use the guiding codebook in Appendix 2 to code other educational media such as books and games for evidence of computational thinking competencies. An additional phase of this research will also include looking at existing engineering curricula for K-2 settings for opportunities to incorporate more computational thinking competencies.

References


Computer Science Teacher Association (CSTA), & International Society for Technology in Education (ISTE). (2011). Computational Thinking Teacher Resources (Second ed.).


Purdue University’s INSPIRE Research Institute for Pre-College Engineering (2017) Computational Thinking Competencies: INSPIRE Definitions. Unpublished resource.


### Appendix 1

**Comparing Computational Thinking Computational Definitions**

<table>
<thead>
<tr>
<th>CT Competencies</th>
<th>Inspire Definitions</th>
<th>Google Definitions</th>
<th>CSTA Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstraction</strong></td>
<td>Identify and utilize the structure of concepts/main ideas.</td>
<td>Identifying and extracting relevant information to define main idea(s)</td>
<td>Reducing complexity to define main idea</td>
</tr>
<tr>
<td><strong>Algorithms and Procedures</strong></td>
<td>Following, identifying, using, and creating an ordered set of instructions. (ie, through selection, iteration and recursion)</td>
<td>Creating an ordered series of instructions for solving similar problems or for doing a task</td>
<td>Series of ordered steps taken to solve a problem or achieve some end.</td>
</tr>
<tr>
<td><strong>Automation</strong></td>
<td>Assigning appropriated set of tasks to be done repetitively by computers</td>
<td>Having computers or machines do repetitive tasks</td>
<td>Having computers or machines do repetitive or tedious tasks.</td>
</tr>
<tr>
<td><strong>Data Collection</strong></td>
<td>Gathering information pertinent to solve the problem</td>
<td>Gathering information</td>
<td>The process of gathering appropriate information</td>
</tr>
<tr>
<td><strong>Data Analysis</strong></td>
<td>Making sense of data by identifying trends</td>
<td>Making sense of data by finding patterns or developing insights.</td>
<td>Making sense of data, finding patterns, and drawing conclusions</td>
</tr>
<tr>
<td><strong>Data Representation</strong></td>
<td>Organizing and depicting data in appropriate ways to demonstrate relationships among data points via representations such as graphs, charts, words or images</td>
<td>Depicting and organizing data in appropriate graphs, charts, words, or images</td>
<td>Depicting and organizing data in appropriate graphs, charts, words, or images</td>
</tr>
<tr>
<td><strong>Debugging/Trouble Shooting</strong></td>
<td>Identifying and systematically addressing problems that inhibit progress toward task completion</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Problem Decomposition</strong></td>
<td>Breaking down data, processes or problems into smaller more manageable</td>
<td>Breaking down data, processes, or problems into</td>
<td>Breaking down tasks into</td>
</tr>
<tr>
<td>Component to be used to solve a problem</td>
<td>Smaller, manageable parts</td>
<td>Smaller, manageable parts</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Parallelization</strong></td>
<td>Simultaneous processing of smaller tasks to more efficiently reach a goal</td>
<td>Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal</td>
<td>Organize resources to simultaneously carry out tasks to reach a common goal.</td>
</tr>
<tr>
<td><strong>Simulations</strong></td>
<td>Developing a model or representation to imitate natural and artificial processes</td>
<td>Developing a model to imitate real-world processes</td>
<td>Representation or model of a process. Simulation also involves running experiments using models.</td>
</tr>
<tr>
<td><strong>Pattern Recognition</strong></td>
<td>Observing patterns, trends and regularities in data (Google)</td>
<td>Observing patterns, trends and regularities in data</td>
<td></td>
</tr>
</tbody>
</table>
Appendix 2.

CT Guiding Codebook

<table>
<thead>
<tr>
<th>CT competencies</th>
<th>Definition</th>
<th>Examples</th>
<th>Non-Examples</th>
<th>Exemplary Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>Identify and utilize the structure of concepts/main ideas.</td>
<td>A player abstracts the function of different commands based on what s/he sees in the real world. (upside arrow means jump, forward arrow means move)</td>
<td>If the app had explicitly explained what each command does to the player without requiring them to use the commands to understand how they work.</td>
<td>The Foos</td>
</tr>
<tr>
<td>Algorithms and Procedures</td>
<td>Following, identifying, using, and creating an ordered set of instructions. (ie, through selection, iteration and recursion)</td>
<td>A player creating series of commands to help the police Foos to get to the stolen doughnut.</td>
<td>If the app awarded progress to players who had randomly chosen a sequence of commands inappropriate to the problem (helping the police Foos to find the doughnut)</td>
<td>The Foos</td>
</tr>
<tr>
<td>Automation</td>
<td>Assigning appropriated set of tasks to be done repetitively by computers</td>
<td>Running a program that includes a series of functions in a loop in which the robot can repeat an act several times.</td>
<td>Repeating a series of commands by manually restarting the function.</td>
<td>DotDash Blocky</td>
</tr>
<tr>
<td>Data Collection</td>
<td>Gathering information pertinent to solve the problem</td>
<td>The robot performs a command; the player collects data from that performance to learn what each command does.</td>
<td>No useful information can be gained from watching a robot performance.</td>
<td>DotDash Blocky</td>
</tr>
<tr>
<td><strong>Data Analysis</strong></td>
<td>Making sense of data by identifying trends</td>
<td>A player finds patterns in the robot performance and associates the performance to the series of commands that were selected.</td>
<td>Data is organized and presented to the player in a way that makes patterns and insights obvious</td>
<td>DotDash Blocky</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------</td>
<td>-----------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Data Representation</strong></td>
<td>Organizing and depicting data in appropriate ways to demonstrate relationships among data points via representations such as graphs, charts, words or images</td>
<td>Understanding progress through the game via leveling maps.</td>
<td>Solving a problem using the collected information, without being ask to form/organize that information into any representations.</td>
<td>The Foos</td>
</tr>
<tr>
<td><strong>Debugging/Troubleshooting</strong></td>
<td>Identifying and systematically addressing problems that inhibit progress toward task completion</td>
<td>In order to complete a level, the player must continue refining his list of commands until the appropriate goal has been reached. Stars encourage players to go back and debug by illustrating the quality solution selected by the player</td>
<td>Instead of finding and fixing the problem, the player resets the level, and creates a new sets of instructions. A player finds/creates a solution and wins the Ninja Foos levels of game. Even if s/he hasn’t gained all the stars, s/he goes to the next level and gets involved in solving the new task.</td>
<td>The Foos</td>
</tr>
<tr>
<td><strong>Problem Decomposition</strong></td>
<td>Breaking down data, processes or problems into smaller more manageable component</td>
<td>To progress through the game, the player has to decide which series of commands (ie, jumping, walking, etc) will help</td>
<td>If the app broke down all the steps necessary to solve the problem and all the player needs to do is select appropriate command</td>
<td>The Foos</td>
</tr>
<tr>
<td>to be used to solve a problem</td>
<td>the police reach the doughnuts</td>
<td>(In the first levels of the Foos and in higher levels of Ninja Foos, the player does not need to break any task down, since the task only requires one command.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Parallelization</strong></td>
<td>Simultaneous processing of smaller tasks to more efficiently reach a goal</td>
<td>A player writes two small programs and combines them at the end so that simultaneously a robot dance and sing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A player runs two different programs sequentially instead of in parallel, so the robot first moves forward and then dances.</td>
<td>DotDash Blocky</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Simulations</strong></td>
<td>Developing a model or representation to imitate natural and artificial processes</td>
<td>The robot play music after giving them a series of commands.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DotDash Blocky</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pattern Recognition</strong></td>
<td>Observing patterns, trends and regularities in data (Google)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>