How Different Input and Output Modalities Support Coding as a Problem-Solving Process for Children

Kening Zhu
City University of Hong Kong
keninzhu@cityu.edu.hk

Xiaojuan Ma
Hong Kong University of Science & Technology
mxj@cse.ust.hk

Gary Ka Wai Wong
Hong Kong Institute of Education
wongkawai@ied.edu.hk

John Man Ho Huen
Koding Kingdom (Hong Kong) Ltd.
john.huen@kodingkingdom.com

ABSTRACT
Using coding education to promote computational thinking and nurture problem-solving skills in children has become an emerging global trend. However, how different input and output modalities in coding tools affect coding as a problem-solving process remains unclear. Of interest are the advantages and disadvantages of graphical and tangible interfaces for teaching coding to children. We conducted four kids coding workshops to study how different input and output methods in coding affected the problem-solving process and class dynamics. Results revealed that graphical input could keep children focused on problem solving better than tangible input, but it was less provocative for class discussion. Tangible output supported better schema construction and casual reasoning and promoted more active class engagement than graphical output but offered less affordance for analogical comparison among problems. We also derived insights for designing new tools and teaching methods for kids coding.

Author Keywords
Graphical; Tangible; Kids coding; Problem solving.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces. K.3.1 [Computers and Education]: Computer Uses in Education.

INTRODUCTION
Educators and researchers have identified the benefits of learning computer programming (a.k.a. coding) at a young age. Papert envisioned that learning coding could help children overcome fear of math, and help them to learn actively [24]. This insight has been later emphasized and proven by other researchers. Clements et al. found that children who learned programming outperformed those who did not in reflectivity, divergent thinking, and metacognitive abilities [7]. Strand et al. [34] reported that programming facilitated collaboration among students, improved their social skills, and encouraged greater focus on their work. As the inventors of Scratch [20] and ScratchJr [10], Resnick et al. [28] argued that through coding, children can develop "computational thinking" [37], which could cultivate creativity and important problem-solving strategies. Solomon [32] suggested that "computer programming can be a useful, creative, and thoroughly entertaining second language for students at all levels." More recently, Wong et al. [38] investigated the impact of coding education at primary schools in Hong Kong, and reported an improvement in the overall performance of the students in mathematics, and the development of soft skills, after learning coding.

The revealed benefits of coding education has motivated many research efforts on new coding tools for children as an alternative to the conventional textual programming environment. The concept of visual programming [4] uses graphical symbols to represent coding concepts and allows children to compose computer programs by piecing graphic icons together. Although textual coding has its own benefits of low viscosity and high expandability [11], visual programming, especially block building, has significant advantages over textual mode for children just entering the venue of computing. It allows children to focus more on learning the computational concepts rather than the complex syntax. As the tangible user interfaces (TUIs) emerged, researchers have also developed tangible block-based programming tools (e.g., Tern [12]). Studies show that graphical-user-interface-based visual programming can achieve better independence in study and learning outcome, while tangible programming is easier to use, more inviting, and more supportive for collaboration [1, 14, 29].

In spite of many research efforts on children's programming education, existing work has mostly emphasized comparing user experience (e.g., the usability of coding interface) with different interface modalities. For example, Horn et al. [13] suggested that children might have difficulty in operating the mouse on a graphical user interface. The emergence of touch surface technology has removed such barriers. Prior studies of hybrid coding interfaces advocated for providing users with the flexibility to choose a preferred modality [13], but the impact of different systems on learning dynamics (e.g., peer interaction and interaction between teachers and students) is largely overlooked. Oviatt's research [23] revealed the impacts of different interfaces on ideation and problem

IDC '16, June 21 - 24, 2016, Manchester, United Kingdom

ACM 978-1-4503-4313-8/16/06□$15.00
DOI: http://dx.doi.org/10.1145/2930674.2930697

(a) Graphical Input (GI) **(b) Graphical Output (GO)** **(c) Tangible Input (TI)** **(d) Tangible Output (TO)**

Figure 1: Different input and output modalities.

(a) GIGO **(b) GITO** **(c) TIGO** **(d) TITO**

Figure 1: Kids coding workshops with different combinations of input and output modalities.

solving for high school and university students, but how different systems would affect young children in learning computational thinking and problem solving is still unclear. A recent study [31] compared children's performance with graphical input and tangible input for robotics, and it suggested fewer programming errors occurred and better debugging was achieved with tangible input. This research examined different input methods for one unified output presentation (a physical robot). However, there was no detailed discussion on the potential affordances of different embodiment (graphical/tangible) of the problem itself, which could affect the problem-solving process [8]. In other words, there still lacks investigation on the affordance of different combinations of input and output modalities on passing problem-solving-related knowledge to children.

In this paper, we explore whether different variations of input and output combinations of coding tools for children (graphical input + graphical output, graphical input + tangible output, tangible input + graphical output, tangible input + tangible output) can adequately support the process of learning and problem solving. Based on the classification by Brown and Chandrasekaran [5], we conceptualized the act of computer programing as one kind of *design-problem-solving* process, where the final goal is clearly known but the problem is ill-structured and open-ended with unclear decomposition plans. Such a process requires important cognitive skills, including problem schema construction, analogical comparison among problems, casual reasoning, and argumentation [17]. We aimed to address the following research questions via four kids-coding workshops:

- *How do different modalities of input and output in coding support the practice of different problem-solving skills for children?*
- *How do different modalities of input and output affect classroom dynamics in coding education, including engagement, attention, peer interaction, and interaction with a tutor?*

The results revealed the advantages and disadvantages of each combination of input and output modalities. We also derived insights for designing new tools and teaching methods for kids coding, leveraging the advantages and eliminating the disadvantages of the different modalities.

KIDS CODING STUDY DESIGN

We conducted a between-subject study via four kids coding workshops, each corresponding to a specific input-output condition (Figure 1).

Design of Kids Coding Tools

We collaborated with a professional kids-coding education organization and customized the scenario of maze solving from the block-based programming library, Blockly [2], with the story of McDull [32], a popular story among local kids, since children would be more engaged with new knowledge in a familiar context than an alien context [27].

The learning objectives included three basic programming concepts (sequence, iteration, and condition) and four problem-solving skills (problem schema construction, analogical comparison, casual reasoning, and argumentation) [17]. The graphical input consisted of a set of virtual block commands (Figure 1a) adapted from Blockly, including movement commands (forward, left, and right), loop commands, and condition commands. We used animation for graphical output (Figure 1b). For tangible input, we wrapped a set of Lego blocks [18] with printed markers to represent the tangible commands (Figure 1c) corresponding with the virtual commands. To compose a program physically, children stacked the blocks together and placed them in front of a web camera. The physical commands can then be recognized and parsed by the computer, and compiled into executable codes. A Roomba cleaning robot [15], which received action commands through Bluetooth connection and moved accordingly on a printed maze, served as the tangible output (Figure 1d). In addition, all the coding tools logged the time stamps for program execution and the program structures.

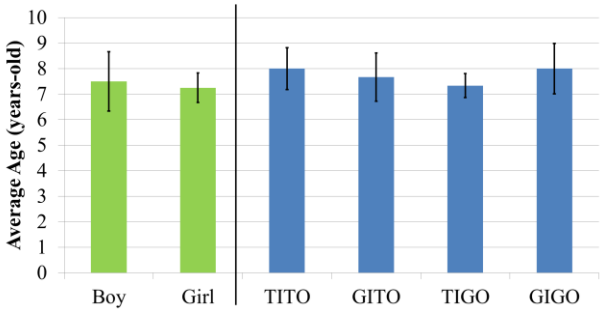

Figure 2: Age distribution of children

Participants

Although there is no clear directive on the appropriate age to learn coding, we decided to focus on the lower elementary stage (i.e., 7 - 10 year-old) [22], a critical period for the development of logic reasoning [26] and second language acquisition [16]. Through word of mouth and the online registration advertised in two local primary schools, within one week, we recruited 12 children (all Chinese, eight male) within this age range (Mean = 7.63, SD = 1.06). The age distribution across gender is shown in Figure 2. According to the pre-workshop questionnaire, the children had never tried coding before but had some experience with computers in studying and gaming.

We invited four experienced tutors from the collaborative organization, and each of them instructed one workshop. All tutors were trained under the same pedagogical scheme based on constructivism [9] and Bloom's taxonomy [3], and had more than six months of experience in teaching visual programming to children from 7 to 10 years old. We briefed all tutors together to familiarize them with all teaching materials and tools before the workshops, and we provided the same course script for all tutors to minimize any individual differences in delivery.

Workshop Design

As letting children freely choose a particular setup would cause unfair comparison on problem-solving performance, we used a between-subject design and assigned the children to four coding workshops that corresponded to the four combinations of input and output modalities: GIGO (Graphical Input Graphical Output), TITO (Tangible Input Tangible Output), GITO (Graphical Input Tangible Output), TIGO (Tangible Input Graphical Output). Note that we assigned 2 boys and 1 girl for each workshop, to minimize the gender effect. With the fixed gender structure, children were randomly assigned to each workshop. The age distribution across four workshops is illustrated in Figure 2.

Considering the class structures of the local elementary education and the suggestions from the collaborative organization, we designed each workshop as a group lesson with three children and a tutor (Figure 1), consisting of three 45-minute sessions with two 15-minute breaks. The four workshops were conducted in parallel at the same time to avoid potential communication among workshops, which may possibly "spoil" the content.

Procedure

After receiving parents' consensuses before the workshops, we collected background information on each child. As the class began, the tutor first presented the background story and motivated the children to help the cartoon character solve the maze. The tutor then taught the three coding concepts (sequence, iteration, and condition) in order. Each concept was taught with two guided examples followed by an exercise which consists of two activities for children to complete. In the first activity, they watched two videos of the cartoon character moving along a pre-defined path. The task was then to program the character to reproduce the same movements, which tested the students' performance on the near transfer of the coding knowledge. The second activity required the children to interpret the movements controlled by two given pieces of codes, and asked them to fix the codes if they were not correct for the given mazes (debugging). Once a participant felt confident with his/her solution, he/she was invited to the front to present it to the rest of the class. All children took turns in the show-and-tell session. They were allowed to go back and correct their code once error was detected during the show-and-tell. At the end of the workshop, all the children, assisted by the tutors, filled out a post-study questionnaire (based on Fun Toolkit [27]) regarding their perception and feelings of coding. The tutors also rated the performance of each student and reflected on their teaching experiences. We took notes of and video recorded each workshop.

ANALYSIS AND RESULTS

In this section, we present findings from the analysis of system logs, video, and questionnaire data, exploring how different modalities affected the classroom dynamics and the development of problem-solving skills in the workshops.

Problem Solving Performance & Engagement

Number of Attempts before Success for Each Exercise

There were a total of three exercises completed by each participant during the workshops. We counted the number of attempts of each child for each exercise (i.e. programs composed and presented before reaching a correct solution) based on the system logs and the captured videos. On average, it took the children in the GIGO workshop the fewest attempts to finish all the exercises (1.3); GITO having the second fewest (2.8), as shown in Figure 3. The children in TITO required a high number of attempts (3.4), although previous studies showed that tangible interface could make coding easier to learn. Mann-Whitely U Test showed that the number of attempts with GIGO was significantly less than the number of attempts with TITO ($U = 6$, $p<0.05$), marginally less than the number of attempts with TIGO ($U = 19.5$, $p = 0.06$), and that there was no significant difference between GIGO and GITO. We also compared the time spent for each exercise based on the system logs, and the analysis revealed the same statistical results as the number of attempts. To ascertain the cause of the differences in performance, we analyzed students' engagement and attention based on the video recordings.

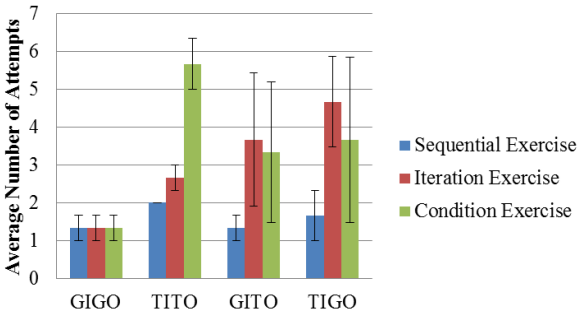

Figure 3: Average number of attempts

Classroom Dynamics: Engagement and Attention

During the GIGO session, children mostly sat at their desk, focusing on their own screens, and rarely responded to the tutors. They occasionally peeked on each other's laptops but seldom discussed the solutions. Students even continued working when others were presenting on the projection screen. The students in the TITO session, however, behaved completely differently. They ran around the classroom, playing on the desks, chairs, and floor. In the beginning of the lesson, they actively responded to the tutor, walked on the physical maze, and watched others' presentations. But they soon lost focus on the class and started to build arbitrary structures with the Lego blocks instead.

On the other hand, the children were less distracted and more responsive to the tutor throughout the entire GITO session. We found that the robot quickly grabbed the children's attention, and kept them engaged with the maze-solving tasks. The children in GITO often walked around the maze, touched and followed the robot, and talked to each other actively. All the kids gathered on the floor and celebrated every successful solution during show-and-tell. The tutor with GITO said, "*It seems the children [established] a close emotional connection with the robot.*" This comment was in line with previous research that suggested users often show more empathy to physical objects than virtual objects [6]. We also observed that children touched the robot and talked to it, such as verbally encouraging the robot to move faster. In addition, the tutor for TITO mentioned that the tangible output could create a stronger sense of achievement when the children solved the physical maze successfully. The children with TIGO also ran around the room frequently, but they were less attentive to the tutor and talked more among themselves.

In summary, we found that tangible input kept children active and encouraged communication in the workshops, but it can be highly distracting as children played with the physical blocks as toys. On the output side, the robot better evoked children's interests with more physical interaction than the virtual animation. We further discuss the performance of different inputs and outputs on cultivating problem-solving skills.

Development of Problem-Solving Skills

We studied the sub-processes of problem solving in detail by analyzing the workshop videos.

Problem Schema Identification/Construction

Problem schema, as defined in [17], consists of structural properties and situational properties of the problem. In our case, the situational properties included the starting position and orientation of the character/robot in the maze, the goal, and the type and number of available blocks. The structural property refers to the computational concepts to be used (sequential, loop, and conditional).

Children in all the workshops had no difficulty identifying the starting position and the goal for all the mazes. By identifying the numbers of available blocks, they also understood the limitations in certain mazes before initiating the coding. However, children with graphical input tended to ignore this limitation during solving the mazes, while children with tangible blocks were more aware of the availability of the resources. This could have been due to the physical reduction of the Lego blocks. This also indicates the necessity of adding more explicit notification on the constraints for graphical coding interfaces.

To apply the appropriate computational concept when coding, one needs to first identify the pattern(s) in the structure of the maze. Before starting block building, children with tangible output tended to walk on the maps trying to understand the directional changes on the path. As stated by Piaget, embodied exploration facilitates schema construction in early childhood [25]. Tangible output serves this purpose and reduces the mental effort for schema construction, making it easier to understand the pattern of the maze, obtain proper structural information, and further identify the optimal concept to use given the constraints. In contrast, children with graphical output spent more mental efforts for orienting on the map. Since they could not physically manipulate the virtual 2D maze, children in GIGO and TIGO groups made mistakes on the turning directions more frequently than those in tangible-output workshops (TITO and GITO). The GIGO tutor noted that some mazes were too abstract for the children, and we observed that children with graphical output often tilted their heads to view the virtual mazes from different angles.

Analogical Comparison among Problems

The analogical comparison and transfer among problems is highly related to the construction of robust problem schema. Following the guided examples of each coding concept, the children were encouraged to solve a new maze by themselves using the taught commands. The graphical and the tangible mazes offered different affordances on problem comparison. The tangible output facilitated better individual schema construction with embodied exploration, but it still required a great amount of mental effort to compare the current maze with the previous mazes on the floor due to the space limitations that prevented two mazes from being shown at the same time. This suggested that transforming the floor-based physical output to a desktop size may provide better support for problem comparison, but this would trade off the facilitation on the embodied schema

	Attention	Engagement	Schema Construction	Analogical Comparison	Casual Reasoning	Argumentation (peer interaction)
GIGO	Highly focused	Not actively communicate	Hard to orient	Compare problems side-by-side	Hard to predict the results	Not keen to present
TIGO	Totally distracted	Talk about unrelated topics				
GITO	Highly focused	Actively engaged	Embodied exploration	Hard to refer to previous problems	Predict by walking on the maze	Actively compete to present solutions
TITO	Occasionally distracted					

Table 1: Comparison of different input/output modalities in coding interfaces for problem solving (white=advantage; grey= disadvantage).

construction. On the other hand, children with graphical maps on the screen often asked the teacher to switch back and forth or open two maps side by side for comparison, making it easier to identify the similarities among problems.

Causal Reasoning

Causal reasoning in problem solving refers to the ability to establish the cause-result relationship within a set of conditions and a set of consequences. As claimed in [17], uncovering causal relationships embedded in problems facilitates predictions. The debugging activity was specifically designed to assess the ability of causal reasoning. We observed that children with tangible output preferred to follow the given commands and walk along the maze while testing the codes. They could predict the movements correctly once they understood the associated programming concepts. In comparison, kids with graphical output had more difficulty in decoding the pre-composed programs in the debugging activity, often tilting their bodies on the seat to figure out the right direction, as it is more difficult to orient in a 2D maze. The tutor for TITO provided the highest rating on the students' ability to recognize bugs in their program (3.67/5) and to correct errors (3.67/5), while the lowest average ratings for these two statements occurred in the GIGO session (3.3/5 and 2.67/5, respectively). The statement "I know how to correct errors" received the highest rating (5/5) from all the children in TITO and the lowest rating (2.3/5) from TIGO.

Argumentation

Argumentation in problem solving includes presenting the solutions then discussing and debating between the problem solvers. Both Piagetian theories on social-arbitrary knowledge [25] and Vygotsky's ZPD (zone of proximal development) theory [36] suggest collaborative activity among children promotes the growth of cognitive skills. Peer interaction is also essential to logical and mathematical thought development [21].

In our study, we were specifically interested in how the children shared their solutions, discussed and learned from one another, and iteratively improved upon their solutions. As discussed previously, children in the GIGO workshop rarely talked to each other, while those in TITO were more active and responsive to the tutor. More specifically, children in GIGO were not keen to share their ideas or to look at each other's results in the show-and-tell portions. They were not eager to compete for the first place either; instead, they worked quietly on their own ideas. The TIGO workshop showed active atmosphere in the room, but the children tended to chat on unrelated topics, such as building arbitrary shapes. In contrast, when the tutors asked "Who wants to demonstrate his/her solution?", all the kids in the sessions with tangible output (GITO & TITO) raised their hands and even competed to run to the front of the classroom. The children with GIGO provided the lowest average rating for their enjoyment of learning with peers (3/5), while all children in other workshops rated 5/5.

In summary (see Table 1), graphical input helped to keep students focused on problem solving better than tangible input but did not enliven the classroom. Tangible output allowed students to physically explore the problem space and attracted students to share their solutions, which could compensate for the disadvantages in graphical input. However, students found difficulties in analogical comparison using floor-based tangible output, while graphical output provided affordances for problem comparison and transfer.

DISCUSSION

Children's Perception on Visual Programming

Results of the post-workshop survey suggested that the children in TITO rated themselves with better understanding of coding (4.67/5), higher confidence (5/5), and greater enjoyment (5/5). This was not the case for those children in TIGO. Details of the ratings of the key questionnaire statements are shown in Figure 4.

Previous research [19] has shown that there is a missing mental link between block-based visual programming and computer science, which may cause low confidence and less interest in coding. Our surveys and interviews further suggested that this phenomenon is particularly significant with TIGO. More specifically, when asked "what is programming", only one student from GIGO session provided a confident answer: "*It is controlling the character's movement*". Most students from GITO and TITO stressed that "*it [coding] was like playing computer games*". In contrast, one 10-year-old child from TIGO workshop clearly stated, "*This is not programming*", which might also reflect the other two students' perceptions. This

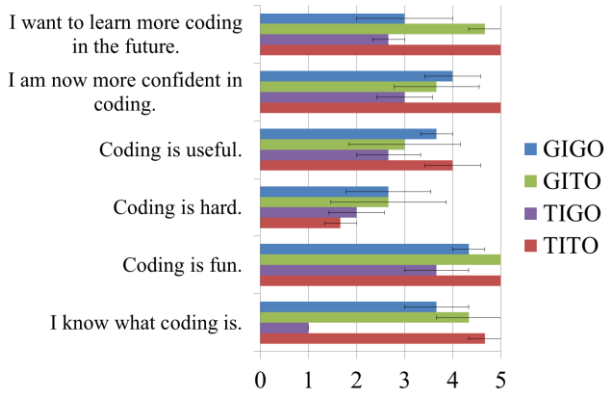

Figure 4: Score distribution for selected questions in children's post-workshop questionnaire

missing connection is also shown in the subjective ratings (Figure 4), as children with TIGO gave lowest rating on their understanding of coding, confidence, and interest for future learning. Further investigation is needed for revealing the possible reasons for why the problem is more severe in TIGO.

Tutors' Reflections and Suggestions

The tutors provided important suggestions for teaching kids coding with different systems and designing a new kids coding platform. The tutor for GIGO mentioned that he was not sure whether the kids actually understood the coding concepts because they seldom reacted to the tutor. Hence, he suggested that adding physical reward as the output for the students and introducing tangible elements into the system could encourage offline communication when using GIGO in classroom, which indicates the potential benefit of hybrid input/output environment [13] for coding education.

The high engagement with tangible input and output at the beginning of the workshops could have been due to the novelty effect of the Lego blocks and the robot. The tutor for TITO commented that the nature of Lego block gives children the impression of toys, which could be the cause of the distraction. However, the novelty effect of tangible blocks indeed reduced their fear of learning and increased their interest in coding. As the workshop went on and the novelty effect faded, the children felt more emotionally connected with the robot than the animated character, and this emotional bond kept the children excited and engaged in classroom communication. The tutor also pointed that one student in TITO expressed the curiosity on what the robot would do and if he built a random 3D structure. She suggested that it would be beneficial to make the coding system recognize the 3D structure of the blocks, and use this information to represent the computational concept; in other words, to leverage the freedom and the affordance of 3D tangible programming interface in the future.

Mazes on the Floor or on the Screen

Papert stated that there may be no mathematical difference between a mechanical (tangible) turtle and light (graphical) turtle, which suggested these two modalities are isomorphic [24]. Czarnocka [8] further argued that "[i]somorphism ensures cognitive comfort". As a counterargument, our research revealed a cognitive difference in these two different embodiments (graphical and tangible) on facilitating children in the problem-solving process. While tangible output allowed for more embodied exploration and construction of individual schema, it provided less support than screen-based graphical maps for analogical comparison and transfer. This leads to an important inquiry to explore in the future, as to whether an augmented-reality-based map (e.g., projection on the floor) would leverage the advantages of both output methods.

Limitation & Future work

We are aware that it can be difficult for children to digest all the delivered knowledge in three hours, and it is difficult to assess their performance quantitatively with a relatively small sample size. In addition, students' behaviors could vary given a different task design. However, this study provided preliminary proof of feasibility for future studies with larger user groups. As the next step, we plan to work closely with local schools to incorporate long-term coding workshops with consistent tutors into the after-class curriculum, to further explore the effect of input and output modality on the transfer of coding knowledge.

CONCLUSION

In conclusion, we conducted four kids coding workshops to investigate how different combinations of input and output modalities used to teach children coding affected the process of problem solving and class dynamics. The results suggested that the graphical input method is less distracting than the tangible input approach, but it is also less provocative during the argumentation stage. Compared with the graphical counterpart, tangible output could better attract students' attention, support more efficient schema construction and casual reasoning, and promote more active argumentation, but it offered less affordance for analogical comparison among problems. We further obtained important insights on designing new kids-coding tools and teaching methods to leverage the benefits and eliminate the disadvantages of different input and output modalities.

SELECTION AND PARTICIPATION OF CHILDREN

In this study, 12 lower elementary children (age distribution: Min = 7, Max = 9, Mean = 7.63, SD = 1.06), from two local schools in Hong Kong were recruited through the word of mouth and the online registration. Each child came with one parent. Prior to the study, the ethical approval was obtained from the university. The consensuses were collected from parents before the study. Parents and children were told about the aims of the research before the workshops. In addition, each parent received a shopping coupon of 200 Hong Kong Dollars after the workshops.

ACKNOWLEDGEMENT

The work described in this paper was fully supported by the Start-up Grant (Project No. 7200404) from City University of Hong Kong.

REFERENCE

1. Bers, M.U., Horn, M.S. *Tangible programming in early childhood: revisiting developmental assumptions through new technologies*. In *High-tech tots: childhood in a digital world*. I.R. Berson, M.J. Berson, (Ed.) Information Age Publishing, Greenwich, (2009).49–70.
2. Blockly. https://code.google.com/p/blockly/.
3. Bloom, B. S. Taxonomy of educational objectives. Vol. 1. New York: David McKay, 1956.
4. Bragg, S.D. and Driskill, C.G. Diagrammatic–graphical programming languages and DoD-STD-2167A. In *Proc. AUTOTESTCON'94*. (1994), IEEE.
5. Brown, David C., and Chandrasekaran, B. *Design problem solving: knowledge structures and control strategies*. Morgan Kaufmann, 2014.
6. Cheok, A. D., Kok, R. T., Tan, C., Newton Fernando, O. N., Merritt, T., & Sen, J. Y. P. *Empathetic living media.* In *Proc. DIS'07*. ACM, 2008.
7. Clements, D. H. *Effects of Logo and CAI environments on cognition and creativity*. Journal of Educational Psychology 78.4 (1986): 309.
8. Czarnocka, M. *Models and symbolic nature of knowledge.* Herfel et al.(eds.) Theories and Models in Scientific Processes (1995): 27–36.
9. DeVries, R., and Kohlberg, L. *Constructivist early education: Overview and comparison with other programs*. Vol. 1987. Natl Assn for the Education, 1987.
10. Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. *Designing ScratchJr: Support for early childhood learning through computer programming.* In *Proc. IDC'13*. ACM, 2013.
11. Green, T. R. G., and Petre, M. *Usability analysis of visual programming environments: a 'cognitive dimensions' framework.* Journal of Visual Languages & Computing 7.2 (1996): 131 – 174.
12. Horn, M. S., and Jacob, R. JK. *Tangible programming in the classroom with tern*. *CHI'07 EA*. ACM, 2007.
13. Horn, M. S., R. Crouser, R., J. and Marina U.B. Tangible interaction and learning: the case for a hybrid approach. *Personal Ubiquitous Comput*. 16, 4 (2012), 379–389.
14. Horn, M.S., Solovey, E.T., Crouser, R.J. and Jacob, R.J.K. Comparing the use of tangible and graphical programming languages for informal science education. In *Proc. CHI '09*. ACM, (2009), 975–984.
15. iRobot Roomba Vacuum Cleaning Robot. http://www.irobot.com/For-the-Home/Vacuum-Cleaning/Roomba.aspx
16. Johnson, J.S., and Newport, E.L. Critical period effects in second language learning: The influence of maturational state on the acquisition of English as a second language. *Cogn. psych*. 21.1 (1989): 60–99.
17. Jonassen, D.H. *Learning to Solve Problems: An Instructional Design Guide*. Pfeiffer, San Francisco, CA (2004).
18. LEGO. http://www.lego.com
19. Lewis, C., Esper, S., Bhattacharyya, V., Fa-Kaji, N., Dominguez, N., & Schlesinger, A. *Children's perceptions of what counts as a programming language*. Journal of Computing Sciences in Colleges, 29.4 (2014), 123–133.
20. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. The scratch programming language and environment. *Trans Comput Educ.* 10,4 (2010):1–15.
21. Mansfield, H., Pateman, N. A., and Bednarz, N. (Eds.) Mathematics for tomorrow's Young Children. Vol. 16. Springer Science & Business Media, 1996.
22. Montessori, Maria. *The absorbent mind*. Macmillan, 1995.
23. Oviatt, S., Cohen, A., Miller, A., Hodge, K., & Mann, A. *The impact of interface affordances on human ideation, problem solving, and inferential reasoning.* ACM Transactions on Computer-Human Interaction (TOCHI) 19.3 (2012): 22.
24. Papert, S. *Mindstorm*. Basic Book, New York (1980).
25. Piaget, J. *The language and thought of the child*. Psychology Press, 1959.
26. Piaget, Jean. The origins of intelligence in children. Vol. 8. No. 5. New York: International Universities Press, 1952.
27. Pritchard, A. *Ways of learning: Learning theories and learning styles in the classroom*. Routledge, 2013, 17–33.
28. Read, J.C and MacFarlane, S. Using the fun toolkit and other survey methods to gather opinions in child computer interaction. In *Proc. IDC '06*. (2006).81–88.
29. Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K. and Torres, R. Growing up programming: democratizing the creation of dynamic, interactive media. In *Proc. CHI EA '09*. ACM, (2009), 3293–3296.
30. Sapounidis, T. and Demetriadis, S. Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal Ubiquitous Comput*. 17, 8 (2013), 1775–1786.
31. Sapounidis, T., Demetriadis, S., & Stamelos, I. *Evaluating children performance with graphical and tangible robot programming tools*. Personal and Ubiquitous Computing, 19(1) (2015), 225–237.

32. Solomon, J. *Programming as a Second Language.* Learning & Leading with Technology 32.4 (2005): 34–39.
33. Story of McDull. http://www.mcdull.hk/.
34. Strand, E., Gilstad, B., McCollum, P. & Genishi, C. *A Descriptive Study Comparing Preschool and Kindergarten LOGO Interaction*. Paper presented at the meeting of the American Educational Research Assiciation, CA, 1986.
35. Van Gog, T., Post, L. S., Napel, R. J. and Deijkers, L. *Effects of objects' "embodiment" on the acquisition of problem-solving skills through practice or video-based modeling example study*. In Proceedings of the 35th annual conference of the cognitive science society, Cognitive Science Society, Austin, TX, 2013.
36. Vygotsky, L. *Interaction between learning and development.* Readings on the development of children 23.3 (1978): 34–41.
37. Wing, J. M. *Computational thinking*. Communications of the ACM 49.3 (2006): 33–35.
38. Wong, G. K. W., Ching, C. C, Mark, K. P., Tang, J. K. T., Lei, C. U., Cheung, H. Y., & Chui, H. L. *Impact of computational thinking through coding in K-12 education: a pilot study in Hong Kong*. Proceedings of the 11th International Conference on Technology Education in the Asia Pacific Region, Hong Kong, 2015.